VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA OSTRAVA
EKONOMICKÁ FAKULTA

KATEDRA APLIKOVANÉ INFORMATIKY

Návrh a implementace mobilní kalendářní aplikace na platformě Android

Design and Implementation of a Mobile Calendar Application for Android platform

Student: Jan Hošna

Vedoucí bakalářské práce: doc. RNDr. Ivo Martiník, Ph.D.

Ostrava 2018

VŠB - Technická univerzita Ostrava
Ekonomická fakulta
Katedra aplikované informatiky

Zadání bakalářské práce

Student: **Jan Hošna**

Studijní program: B6209 Systémové inženýrství a informatika

Studijní obor: 6209R017 Informatika v ekonomice

Téma: Návrh a implementace mobilní kalendářní aplikace na platformě Android
Design and Implementation of a Mobile Calendar Application for Android platform

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Úvod
2. Teoretické pojmy a východiska tvorby aplikací na platformě Android
3. Analýza současného stavu
4. Návrh a realizace mobilní aplikace
5. Závěr

Seznam použité literatury
Seznam zkratek
Prohlášení o využití výsledků bakalářské práce
Seznam příloh
Přílohy

Seznam doporučené odborné literatury:

BUCHALCEVOVÁ, Alena a Iva STANOVSKÁ. *Příklady modelů analýzy a návrhu aplikace v UML.* Praha: Oeconomica, 2013. ISBN 978-80-245-1922-7.
LACKO, Ľuboslav. *Mistrovství - Android.* Brno: Computer Press, 2017. ISBN 978-80-251-4875-4.
SCHILDT, Herbert. *Java 8: výukový kurz.* Brno: Computer Press, 2016. ISBN 978-80-251-4665-1.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. RNDr. Ivo Martiník, Ph.D.**

Datum zadání: 24.11.2017
Datum odevzdání: 11.05.2018

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA OSTRAVA Ekonomická fakulta 1

Ing. Petr Rozehnal, Ph.D.
vedoucí katedry

prof. Dr. Ing. Zdeněk Zmeškal
děkan fakulty

„Prohlašuji, že jsem celou práci, včetně všech příloh vypracoval samostatně."

V Ostravě dne 11.5.2019 .....

.....

Jan Hošna

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce doc. RNDr. Ivo Martiníkovi, Ph.D. za odborné vedení, cenné rady a připomínky k práci.

Obsah

1 Úvod

Mobilní technologie získávají každým rokem na popularitě a vše nasvědčuje tomu, že tento trend bude pokračovat i nadále. Lidé již méně sledují televizní obrazovku a jejich pozornost se přesunula do oblasti mobilních zařízení. S tím souvisí i změny situace na trhu práce a vznik nových průmyslových odvětví založených na mobilních technologiích.

Aktuálně nejrozšířenějším operačním systémem na mobilních zařízeních je Android. Vlastnictví platformy Android firmou Google jen přispívá ke stabilitě této platformy, jejímu dobrém jménu a rozsáhle podpoře jak v oblasti uživatelské, tak i vývojářské.

Prosperující firmy v IT světě chtějí neustále inovovat, ale zároveň mít vše pod kontrolou, protože jinak by v neustále měnícím se prostředí s velkou pravděpodobností neobstály.

Mezi takové firmy patří i společnost Tieto s.r.o., jejíž zaměstnanci neustále hledají možnosti zlepšení a inovace současného stavu. Společnost Tieto s.r.o. zaměstnává v Ostravě přes 2000 lidí a řeší IT problematiku mnoha svých klientů. Protože má společnost sídlo ve Finsku a další významné pobočky zejména v Indii, dochází často k tomu, že pracovníci nebo klienti převážně z těchto zemí navštěvují ostravskou pobočku.

Organizace těchto návštěv je z důvodu rozsahu společnosti komplikovaná a zejména pro koncového uživatele je aktuální řešení založené na webových technologiích poměrně nepraktické a nemoderní. V současnosti dochází k zasílání základních informací a detailů týkajících se návštěv prostřednictvím elektronické pošty. Již dříve došlo k vytvoření webového rozhraní pro stranu organizační a také již bylo programově vytvořeno řešení zasílání dat o událostech ze strany serveru.

Vzestup popularity mobilních zařízení a snaha o inovace vedla společnost Tieto s.r.o., k vytvoření poptávky po mobilní aplikaci na platformě Android, která by kalendářním způsobem zobrazovala doprovodný program návštěv.

Prvním a hlavním cílem bakalářské práce je analýza, návrh a implementace mobilní aplikace pro mobilní platformy využívající operační systém Android. Aplikace bude zaměřena zejména na přehledné zobrazení doprovodného programu návštěv ve společnosti Tieto s.r.o. kalendářním způsobem, přičemž potřebná data budou získána prostřednictvím komunikace s centrálním serverem provozovaným společností. Velká pozornost bude věnována zejména návrhu a implementaci uživatelsky přátelského rozhraní této mobilní aplikace.

Druhým cílem práce je popis základních teoretických pojmů souvisejících s vývojem aplikací na platformě operačního systému Android.

Třetím cílem práce je popis východisek tvorby aplikací na platformě Android.

Čtvrtým cílem práce je popis nástrojů užívaných při vývoji, jako jsou na trhu dostupná vývojová prostředí, případně emulátory.

Ve druhé kapitole jsou vysvětleny základní pojmy a východiska potřebná pro vývoj aplikací pro platformu operačního systému Android, jeho historie, architektura, základní prvky a jejich životní cyklus, verze operačního systému a jejich vlastnosti, porovnání s konkurenty v oblasti mobilních operačních systémů, nástroje potřebné při vývoji a rovněž popis způsobů, které mohou být využity pro implementaci a distribuci aplikací prostřednictvím služby Google Play. Součástí této kapitoly je i vymezení pojmu softwarová aplikace.

Ve třetí kapitole je popsána společnost, ve které je aplikace vyvíjena, včetně analýzy jejího současného stavu a problému, který je v práci řešen.

Čtvrtá kapitola je věnována návrhu a vývoji aplikace, včetně zvolení způsobu tvorby aplikace a pomocných nástrojů pro vývoj.

Závěr bude věnován zhodnocení využitých technologií pro implementaci dané mobilní aplikace spolu s vyhodnocením výsledků bakalářské práce vůči stanoveným cílům.

2 Teoretické pojmy a východiska tvorby aplikací na platformě Android

2.1 Historie operačního systému Android

Vznik společnosti Android Inc. se uskutečnil v říjnu roku 2003 v kalifornském Palo Alto. Tato společnost se dle jednoho ze zakladatelů Andyho Rubina zpočátku zabývala zlepšením operačního systému digitálních fotoaparátů. Fotoaparát s operačním systémem Android se byl schopen bezdrátově připojit k počítači, pomocí kterého mohlo dojít k zálohování fotografií do cloudového úložiště.

Trh digitálních fotoaparátů byl v té době v poklesu, a proto došlo k rozhodnutí zaměřit se na operační systémy mobilních zařízení. Stejný operační systém, který byl určený pro digitální fotoaparáty, se stal operačním systémem pro mobilní zařízení. (www.androidauthority.com)

Společnost Google tedy operační systém Android nenavrhla ani nevyvinula, ale odkoupila. Stalo se to dva roky po založení společnosti Android Inc., tedy v roce 2005, za částku 50 milionů dolarů. Aktuální hodnota společnosti Android Inc. se udává o tři řády vyšší.

Dodnes stojí za vývojem operačního systému Android Open Handset Alliance, která byla založena v listopadu 2007. V této době vzniká i první vývojářský soubor nástrojů pro vývoj softwaru (SDK).

Dalším významným datem je září 2008, kdy v USA došlo k uvedení na trh telefonu HTC Dream (G1) jakožto prvního telefonu s operačním systémem Android. (Lacko, 2017)

2.2 Architektura Androidu

Operační systém Android se skládá z několika softwarových vrstev, které lze rozdělit do pěti základních. Vrstvy spolu vzájemně komunikují a díky tomu, že zde není pouze jedna vrstva, tak lze abstrahovat od vrstev nižších, a tím nejen ulehčit vývoj aplikací, ale i zajistit přenositelnost mezi zařízeními s tímto operačním systémem.

Jednotlivé vrstvy operačního systému Android:

1. Aplikační
2. Aplikační rámec Programového Rozhraní Java
3. Nativní C/C++ knihovny a Android Runtime
4. Vrstva Hardwarové Abstrakce
5. Linuxové jádro

Pro zvýšení přehlednosti jsou popsány jednotlivé vrstvy od nejnižší, tedy Linuxového jádra až po nejvyšší, Aplikační. (Lacko, 2017)

2.2.1 Linuxové jádro

Základem operačního systému Android a nejnižší vrstvou architektury je linuxové jádro, které je upraveno pro fungování na mobilních zařízeních. Upravení spočívá zejména ve sníženém množství funkcí a jejich přizpůsobení.

Linuxové jádro zabezpečuje interakci s hardwarem a díky této vrstvě je tato interakce abstrahována pro vrstvy vyšší. Tato vrstva zajišťuje například správu procesů, paměti, základní síťovou vrstvu, vstupně-výstupní operace, základní grafiku a napájení. (Lacko, 2017)

Pro řízení komunikace mezi procesy slouží Service Manager, u kterého je pro tyto účely každá spuštěná aplikace zaregistrována. S komunikací mezi procesy mohou být spojena velká bezpečnostní rizika, proto je na platformě Android implementován meziprocesový ovladač volání metod a komunikace zvaný Binder. Binder mapuje reference a s využitím sdílené paměti zprostředkovává údaje pro více procesů, čímž jsou tato rizika ošetřena. (Lacko, 2017)

2.2.2 Vrstva Hardwarové Abstrakce

Typů zařízení s operačním systémem Android je mnoho, proto je nad vrstvou Linuxového jádra i vrstva hardwarové abstrakce. Díky této vrstvě se mohou zařízení dle hardwarové specifikace lišit, avšak bude zajištěno fungování komunikace mezi nejnižší vrstvou a vyššími vrstvami systému.

2.2.3 Nativní C/C++ knihovny a Android Runtime

Nativní knihovny systému Android zabezpečují fungování a přístup ke komponentám tohoto systému. Jedná se o mezivrstvu mezi linuxovým jádrem a vyššími vrstvami, která např. spravuje multidotykový displej, knihovny médií a SQLite databáze.

Android Runtime slouží k přeložení bajtového mezikódu do nativních instrukcí pro mobilní zařízení, respektive pro konkrétní procesor. Bajtový mezikód vzniká komplikací souborů JAR a CLASS a je označován jako formát DEX. Tento formát se využívá zejména z toho důvodu, že kód je oproti souborům CLASS kompaktnější, a to je v oblasti mobilních zařízení velkou výhodou. (Lacko, 2017)

2.2.4 Aplikační rámec Programového Rozhraní Java

Tento rámec je považován pro vývojáře aplikací za nejdůležitější vrstvu. Vrstva je napsána v programovacím jazyce Java a obsahuje například ikony, ovládací prvky a také aplikacím poskytuje základní služby systému. V této vrstvě je spravován životní cyklus

aplikací, správa nainstalovaných balíčků, upozornění, případně jednotlivá okna, ze kterých sestávají aplikace.

2.2.5 Aplikační vrstva

Aplikační vrstva je na samém vrcholu hierarchie architektury operačního systému Android. V této vrstvě jsou obsaženy jednotlivé aplikační programy. Mezi tyto programy řadíme například seznam kontaktů, navigaci nebo program na zasílání zpráv. (Lacko, 2017)

2.3 Verze operačního systému Android

Operační systém Android od svého založení prošel mnoha změnami. Proto si každý vývojář mobilních aplikací pro platformu Android musí předem rozmyslet, na kterou z jeho verzi bude cílit. Z toho vycházejí i možnosti, které pak při vlastním vývoji aplikací má.

Z rozhodnutí, jaká bude minimální podporovaná verze operačního systému Android pro danou vyvíjenou aplikaci vychází i počet potenciálních uživatelů. Protože pouze ti uživatelé, kteří mají operační systém Android na této, případně vyšší verzi, ji mohou využívat.

Zároveň je se zvolením minimální podporované verze spjata i nemožnost využívat programová rozhraní vyšších verzí.

Proto, pokud podmínky vývoje dané aplikace nejsou předem striktně stanoveny, je důležité vybrat takovou verzi operačního systému, aby tuto aplikaci mohlo používat co nejvíce uživatelů, ale bez nutnosti znatelného snížení uživatelského komfortu.

Navíc je nutné si uvědomit, že vývoj postupuje stále kupředu a je třeba myslet na to, jaká bude situace v době, kdy bude aplikace uvedena na trh.

Zajímavostí je, že jednotlivé verze operačního systému Android mají svá jména podle sladkostí. Poslední vydanou verzi představuje verze číslo 8.0 s názvem Oreo. Při výběru minimální podporované verze nás však zajímá verze programového rozhraní. V tabulce 2.1 je znázornění jednotlivých verzí operačního systému s jejími názvy vůči verzi programového rozhraní.

Tabulka 2.1 Historie verzí operačního systému Android

Verze Programového Rozhraní	Název	Verze Android
1	Bez oficiálního názvu	1.0
2	Bez oficiálního názvu	1.1
3, NDK 1	Cupcake	1.5
4, NDK 2	Donut	1.6
5	Eclair	2.0
6	Eclair	2.0.1
7, NDK 3	Eclair	2.1
8, NDK 4	Froyo	2.2.x
9, NDK 5	Gingerbread	2.3-2.3.2
10	Gingerbread	2.3.3-2.3.7
11	Honeycomb	3.0
12, NDK 6	Honeycomb	3.1
13	Honeycomb	3.2.x
14, NDK 7	Ice Cream Sandwich	4.01-4.0.2
15, NDK 8	Ice Cream Sandwich	4.0.3-4.0.4
16	Jelly Bean	4.1.x
17	Jelly Bean	4.2.x
18	Jelly Bean	4.3.x
19	KitKat	4.4–4.4.4
20	KitKat	4.4W
21	Lollipop	5.0
22	Lollipop	5.1
23	Marshamallow	6.0
24	Nougat	7.0
25	Nougat	7.1
26	Oreo	8.0.0
27	Oreo	8.1.0

Zdroj: (www.android.com) – vlastní zpracování

V tabulce 2.1 jsou zobrazeny názvy jednotlivých verzí Android, kromě prvních dvou verzí, které neměly oficiální název, a příslušné verze programového rozhraní spolu s verzí

nativního vývojářského souboru nástrojů pro vývoj softwaru, které jsou značeny NDK. NDK nám dovoluje využívat nejen programovací jazyk Java, případně Kotlin, ale i C a C++, čímž lze využívat knihovny napsané v těchto programovacích jazycích, případně psát aplikace přímo v těchto programovacích jazycích.

Zajímavostí na této tabulce je verze programového rozhraní 20, kdy došlo k počátku vývoje aplikací na nositelnou elektroniku s operačním systémem Android. (www.developer.android.com)

2.4 Základní prvky Android aplikací

Pro vývoj aplikací je nutné znát, z čeho se aplikace skládají, a uvědomit si, jak lze tyto části při vývoji aplikace navzájem kombinovat. Mezi základní součásti aplikací se řadí čtyři prvky, jmenovitě aktivity, služby, vysílače spolu s přijímači a poskytovatelé obsahu. Někdy se k těmto čtyřem základním prvkům řadí i upozornění neboli notifikace. Každý prvek má své využití, avšak v praxi je užitek největší při správné kombinaci všech prvků. (Lacko, 2015)

2.4.1 Aktivity

Aktivita představuje uživatelské rozhraní, které je zobrazeno po spuštění aplikace. Aplikace se může skládat z více aktivit a tyto aktivity pak slouží k interakci mezi uživatelem a aplikací. Jednotlivé interakce mohou být zaznamenávány a využity i mezi aktivitami navzájem. Aktivity tedy slouží k zobrazování uživatelského rozhraní a získávání informací ohledně interakce uživatele s aplikací.

2.4.2 Služby

Pro operace, které trvají déle, případně probíhají na pozadí, se využívají služby. Jako příklad se nejčastěji uvádí přehrávání hudby na pozadí, kdy uživatel pomocí aplikace, která toto umožňuje, spustí hudbu, a poté spustí jinou aplikaci, čímž aplikace, která spustila hudbu, přejde svým během do pozadí. V tomto případě hudba pomocí služby hraje i přes skutečnost, že její běh probíhá na pozadí.

2.4.3 Vysílače a přijímače

Často se stává, že na zařízení dojde k nějaké události. Tato událost představuje objekt typu intent, neboli záměr. Tento záměr může být vyslán do zařízení prostřednictvím vysílačů. Oproti tomu mohou přijímače poslouchat na pozadí na tyto vyslané záměry a pak na ně reagovat. Typickým příkladem je SMS zpráva, kdy skrze aplikaci dojde k vyjádření záměru, na který čeká příjemce této zprávy. Avšak příjemce nemůže vědět, kdy mu bude nějaká SMS zpráva zaslána, respektive kdy dorazí. Proto operační systém poslouchá na pozadí jakožto

přijímač a když přijme záměr o přijetí SMS zprávy, tak spustí službu a zprávu uloží. (Lacko, 2015)

2.4.4 Poskytovatelé obsahu

Pomocí poskytovatelů obsahu lze sdílet data mezi více aplikacemi. Slouží zejména k ochraně údajů, jelikož při implementaci má vývojář kontrolu nad tím, jak k datům v aplikaci mohou přistupovat jiné procesy. (Lacko, 2015)

2.4.5 Upozornění

Mobilní aplikace mají možnost upozorňovat uživatele. Upozornění představují v mobilních aplikacích text a případně ikonu, které se zobrazí na obrazovce mobilního zařízení. Příjem upozornění může být doprovázen i zvukovým efektem.

Upozornění lze dělit dle místa původu, pracuje se s nimi stejným způsobem a uživatel je od sebe nerozezná. Aplikace musejí vždy zažádat o povolení těchto upozornění, aby se zabránilo možnosti, že uživateli budou zasílány upozornění od aplikací, o které nemá zájem.

Typy upozornění:

- **lokální upozornění** vysílají nainstalované aplikace, může se například jednat o upozornění na blížící se událost,
- **vzdálená upozornění** vysílá server a mobilní zařízení s nainstalovanou aplikací připojené k internetu upozornění přijme. Vzdálená upozornění se často využívají pro zasílání komerčních sdělení. (Vávrů, Ujbányai, 2013)

Upozornění se typicky zobrazují třemi způsoby, vždy záleží na vývojáři, který způsob uzná, že se pro danou situaci nejlépe hodí:

Druhy reakcí na upozornění:

- Toast,
- Oznámení ve stavovém řádku,
- Dialogové okno.

Toast

Při použití tohoto způsobu dojde k zobrazení oznámení na pracovní ploše. Oznámení zabírá pouze tolik prostoru, kolik vyžaduje text v oznámení. Na toto oznámení nelze reagovat a automaticky zmizí. Nejčastěji se používá pro oznámení úspěšného uložení souboru, případně chybové hlášky o neúspěchu při ukládání. (Vávrů, Ujbányai, 2013)

Oznámení ve stavovém řádku

Tento způsob zajistí, že do stavového řádku bude přidána zpráva a ikona aplikace, případně může dojít k signalizaci zvukové, světelné, nebo k vibraci zařízení. Tato oznámení

mají využití zejména pokud aplikace běží v pozadí a je záměrem uživatele informovat o nějaké vzniklé události.

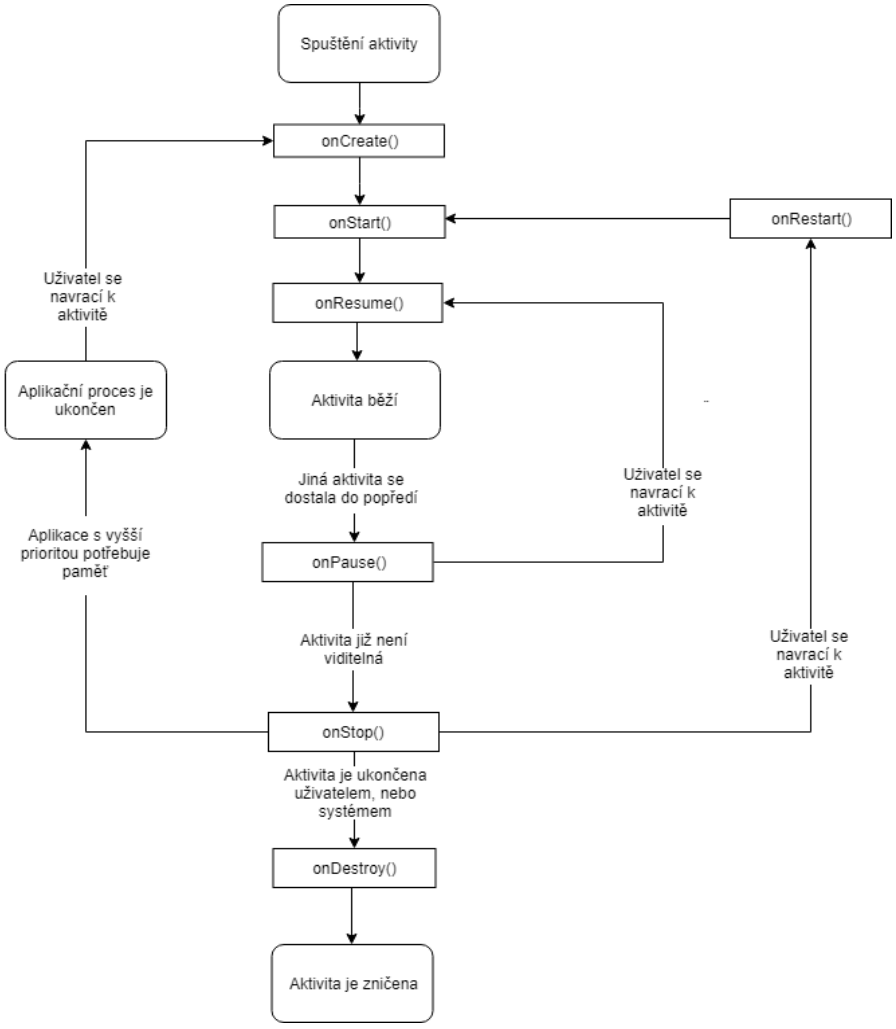
Dialogové okno

Využití oznámení prostřednictví dialogového okna má význam například v případě, kdy uživatel chce smazat nějaký soubor. Uživatel mohl omylem zvolit jiný soubor, než chtěl. Pro tyto případy se před smazáním souboru zobrazí dialogové okno, které vyžaduje potvrzení akce uživatelem stisknutím tlačítka OK, nebo zrušení akce stisknutím tlačítka Storno. (Vávrů, Ujbányai, 2013)

2.5 Životní cyklus aktivity

Co je to aktivita a jaký má význam, již bylo zmíněno výše. Pro vývojáře je však nutné pochopit nejen smysl aktivity, ale i její životní cyklus. V schématu 2.1 je zobrazen životní cyklus aktivity.

Schéma 2.1 Životní cyklus aktivity



Zdroj: (www.developer.android.com) - vlastní zpracování

Jak je vidět v schématu 2.1, po spuštění aktivity dojde k provedení metody `onCreate`, ve které dochází k vytvoření uživatelského rozhraní a konfiguraci proměnných a objektů, potřebných pro běh aplikace. Po provedení metody není aplikace viditelná, ani s ní uživatel nemůže komunikovat. Aby bylo možné s aplikací komunikovat, je třeba provést metodu `onStart`, která slouží k tomu, aby bylo možné zobrazit uživatelské rozhraní a následně metodu `onResume`, která se provádí, když aplikace vstupuje do popředí. (Lacko 2015)

V případě, že se do popředí dostane jiná aktivita, bude nad původní aktivitou provedena metoda `onPause`. Uživatel se k původní aktivitě může navrátit a dále s ní pracovat, případně může dojít k tomu, že aktivita přestává být viditelná. Tehdy bude provedena metoda `onStop` a pokud by se chtěl uživatel k aplikaci vrátit, bude následně provedena metoda `onRestart`.

Pokud aktivita není viditelná, může dojít k jejímu ukončení v důsledku alokace paměti aplikací s vyšší prioritou. Pokud by se chtěl uživatel k původní aplikaci vrátit, bude provedena metoda `onCreate`. (Lacko 2015)

Pokud dojde k ukončení aplikace systémem nebo uživatelem, bude provedena metoda `onDestroy` a dojde k ukončení životního cyklu aktivity.

2.6 Google Play

Google Play poskytuje možnost všem vlastníkům mobilních zařízení s operačním systémem Android objevovat nové aplikace, které si poté mohou stáhnout, v případě placených aplikací nejprve zaplatit.

Je to tedy virtuální trh, na kterém mohou vývojáři nabízet své aplikace všem potenciálním zákazníkům. Z toho vyplývá i obrovská výhoda pro vývojáře, kteří tímto mají možnost získat mnohem více zákazníků, než kdyby se snažili o distribuci svých výtvorů například nabídnutím stažení ze svých internetových stránek.

Každá aplikace, která chce být zařazena do Google Play katalogu, musí splňovat základní požadavky na funkčnost a kvalitu aplikace. Zároveň je nutné souhlasit s podmínkami využívání Google Play. Mezi mnohými dalšími podmínkami je vlastnictví vývojářského účtu pro Google play, který stojí 25 dolarů.

Mezi zajímavosti patří i možnost vkládat privátní aplikace. Tato možnost je velmi lukrativní pro firemní aplikace, u kterých chceme mít distribuci pod kontrolou.

2.7 Softwarové aplikace a informační systémy

Softwarové aplikace, případně aplikační software, nebo zkráceně aplikace je tvořena moduly, objekty, komponenty a službami, které spolu s jejími vzájemnými vazbami tvoří programové vybavení, které užívají jiné aplikace, nebo uživatelé. Tyto aplikace jsou tvořeny za účelem podpory činnosti uživatelů, případně k podpoře byznysu. (Buchalcevová, Stanovská, 2013)

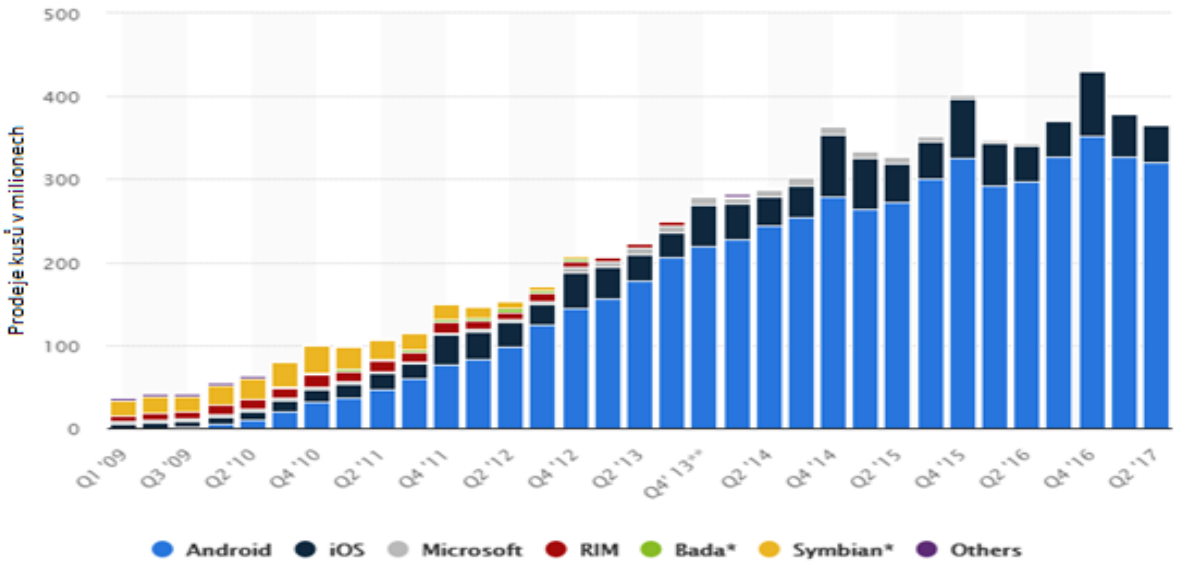
Informační systém je systémem lidí, dat, informačních a komunikačních technologií, který má za cíl podporu procesů na všech úrovních řízení. Jedná se o procesy rozhodovací, řídící a informační. (Buchalcevová, Stanovská, 2013)

Informační systém je širším pojmem než aplikace. Informační systém se z pravidla skládá z více aplikací, které slouží zejména pro podporu automatizovaných činností. (Buchalcevová, Stanovská, 2013)

2.8 Mobilní platformy a jejich srovnání

Před vývojem aplikace je důležité si uvědomit, proč má smysl mobilní aplikace vyvíjet, a proč právě pro platformu Android. Zásadní jsou samozřejmě statistická data. V grafu 2.1 je grafické znázornění zastoupení mobilních operačních systémů na trhu.

Graf 2.1 Grafické znázornění zastoupení mobilních operačních systémů na trhu

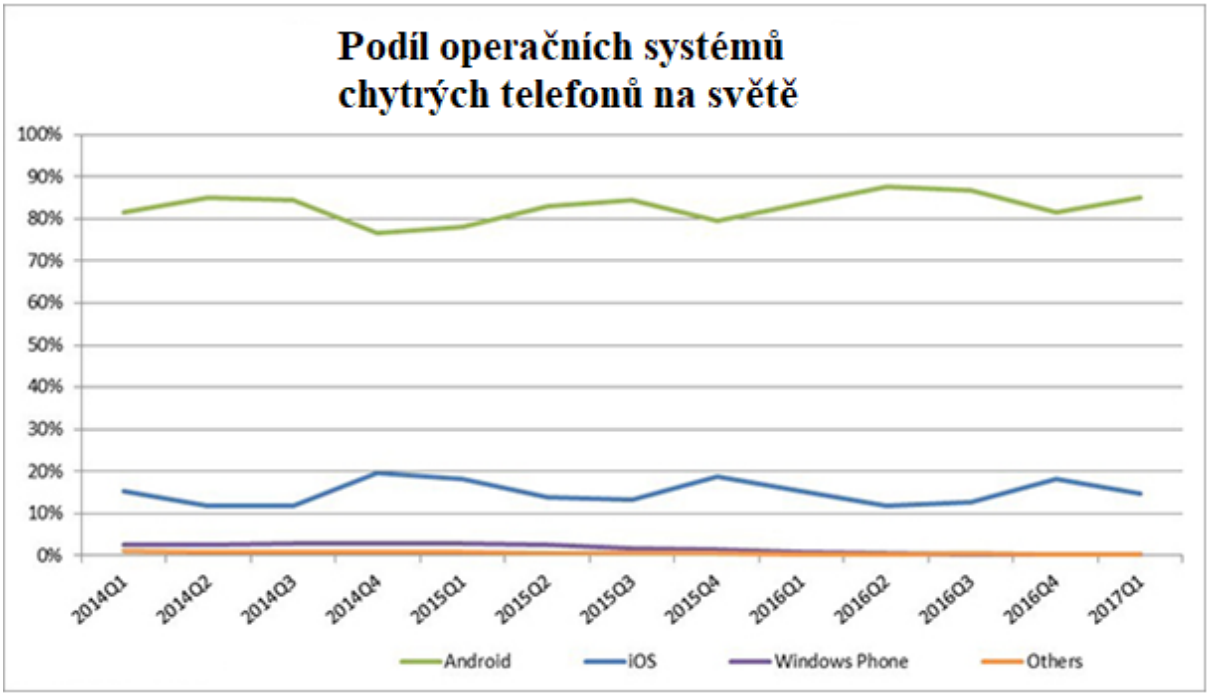


Zdroj: (www.statista.com) – vlastní zpracování

Chytré mobilní telefony jsou v dnešní době velice populární a za rok 2016 bylo prodáno 1.5 bilionů zařízení, jak je vidět na grafu 2.1. Na trhu bylo dříve mnoho konkurentů, kteří chtěli získat co největší podíl na trhu svými operačními systémy. Jak je však vidět na

grafu, tak od roku 2013 jsou v zásadě jedinými konkurenty na trhu Android od Google a iOS od Apple. V grafu 2.2 je zobrazen podíl na trhu jednotlivých operačních systému.

Graf 2.2 Podíl na trhu jednotlivých operačních systému



Zdroj: (www.IDC.com) – vlastní zpracování

V tabulce 2.2 je zobrazený podíl na trhu jednotlivých mobilních operačních systémů.

Tabulka 2.2 Podíl na trhu jednotlivých mobilních operačních systémů

Období	Android	Windows Phone	iOS	Others
2016Q1	83.4 %	0.8 %	15.4 %	0.4 %
2016Q2	87.6 %	0.4 %	11.7 %	0.3 %
2016Q3	86,8 %	0.3 %	12.5 %	0.4 %
2016Q4	81.4 %	0.2 %	18.2 %	0.2 %
2017Q1	85.0 %	0.1 %	14.7 %	0.1 %

Zdroj: (www.IDC.com) – vlastní zpracování

V grafu 2.2 a v tabulce 2.2 je patrné rozdělení trhu mezi zmíněnými dvěma platformami, tedy iOS a Android, včetně zastoupení Windows Phone a ostatních. Z grafu a tabulky je zjevný trend poklesu platforem jiných, než jsou iOS a Android.

Výrobci telefonů, jako například Nokia, která dodávala své telefony s operačním systémem Symbian, z grafu 2.1 viditelně zpočátku jednoznačně nejrozšířenějším, ustoupila od vývoje a telefony dodává s operačním systémem Android. Totéž platí i o firmě Microsoft, která se snažila na trhu zabodovat s operačním systémem Windows Phone, avšak v říjnu roku

2017 viceprezident pro operační systémy v Microsoftu Joe Belfiore oznámil, že Windows Phone nebude pokračovat ve vývoji nových funkcí, jelikož z důvodu příliš nízkého podílu na trhu se firmě Microsoft nevyplatí ve vývoji pokračovat.

2.9 Vývojová prostředí

Vyvíjet aplikace pro Android lze v mnoha vývojových prostředích. Výběr je velmi důležitý, jelikož může při vývoji šetřit mnoho práce. Některé firmy a jejich vývojové týmy mají pevně dané, v jakém vývojovém prostředí pracují, aby zamezili problémům vycházejících z užívání různých prostředí. Proto je důležité mít přehled o trendech a moderních prostředcích.

2.9.1 IntelliJ IDEA

IntelliJ IDEA je vývojové prostředí, které je považováno za nejmodernější prostředí pro vývoj aplikací zejména v jazyce Java. Vyznačuje se zejména skvělou podporou vývojářů pomocí napovídání v jakémkoliv kontextu a chytré doplňování kódu.

Poskytuje také kvalitní nástroje pro refaktorování kódu a významným bonusem je velká podpora pro práci s frameworky, například pro Java framework Spring. (www.jetbrains.com)

2.9.2 Android Studio

Toto vývojové prostředí bylo vytvořeno na základech IntelliJ IDEA. Je oficiálně podporováno Googlem v oblasti vývoje aplikací pro platformu Android. Je to vývojové prostředí navrženo na míru pro vývoj Android aplikací.

Android studio nabízí předlohy a ukázkové aplikace, které slouží začínajícím programátorům k inspiraci, případně i zkušenějším k ušetření času. Podporuje také práci s testovacími nástroji a frameworky, jako je například JUnit 4.

Layout editor je další snahou k ulehčení práce vývojáře. Usnadňuje totiž práci při návrhu aplikace, kdy vývojář může z předem vytvořených komponent technikou přetahování vytvořit celkový vzhled aplikace a poté upravit dle svých potřeb.

Za zmínku rovněž stojí i pomoc vývojového prostředí s integrací Cloudových řešení. Android studio má k dispozici Firebase Asistenta, který poskytuje možnost připojení mobilní aplikace k využívání analýz, autentizace, vzdálených upozornění poskytovaných společností Google, prostřednictvím platformy Firebase. (www.developer.android.com)

2.9.3 Eclipse

Eclipse, jakožto vývojové prostředí se zaměřuje zejména na Javu, ale lze v něm vyvíjet i například v C/C++, Python a další. Eclipse je zdarma ke stažení a užívání. Navíc lze rozšiřovat možnosti využití pomocí různých plug-inů. (www.tutorialspoint.com)

2.9.4 WebStorm

WebStorm je rovněž vývojové prostředí založené na základu IntelliJ IDEA. WebStorm je na rozdíl oproti ostatním zmíněným variantám soustředěn zejména na programovací jazyk JavaScript a technologie využívané k programování strany klienta.

Využití má zejména v případě práce s nejmodernějšími frameworky jako jsou React, Vue a Angular. V tomto prostředí lze vyvíjet nejen desktopové aplikace, ale i mobilní aplikace a aplikace na straně serveru s využitím frameworku Node.js. (www.jetbrains.com)

2.10 Emulátory

Během vývoje mobilních aplikací je třeba aplikaci testovat na výsledných zařízeních. Toto lze zajistit dvěma způsoby. Buď na reálném zařízení, nebo pomocí virtuálního zařízení, tedy emulátoru.

Reálná zařízení mají velkou výhodu v tom, že na nich lze vyzkoušet, jak se bude s aplikací pracovat z pozice koncového uživatele. Ale jsou zde jistá omezení. Pokud by bylo potřeba vyzkoušet, jak funguje responsivní design aplikace, bylo by nutné zajistit mnoho zařízení, což by bylo nákladné a nepraktické. Proto lze využít virtuálních zařízení, která dokáží simulovat zařízení s různým rozlišením displeje, velikostí displeje, výkonem, nebo verzí operačního systému.

2.10.1 Android Studio Emulátor

Android Studio poskytuje vlastní emulátor. Podpora tohoto vývojového prostředí se přenesla i na tento emulátor, jelikož práce s nastavením a spuštěním je jednoduchá a intuitivní. Pomocí tohoto emulátoru lze nastavit mnoho prvků, například velikost displeje, datového úložiště, operační paměti, také je možné měnit rozlišení displeje. Funguje zde i například přední a zadní kamera. (www.developer.android.com)

2.10.2 Genymotion

Tento emulátor patří mezi profesionálními vývojáři k nejoblíbenějším, avšak každý začínající vývojář v oblasti vývoje aplikací pro platformu Androidu si k němu musí najít cestu, jelikož nemá takovou mediální podporu, jakou má emulátor Android Studia zmíněný výše.

Emulátor je jako stvořený pro účely vývoje Android aplikací, jelikož je oproti ostatním výkonnější a stabilnější. Nestává se u něj, že by přestal fungovat, a poskytuje veškeré funkce, které jsou při testování aplikace potřeba. Nevýhodou je, že pro komerční užití je placený a nejlevnější plán s využitím pro jednoho uživatele stojí 99 euro na rok. (www.genymotion.com)

2.10.3 Visual Studio Emulátor

Tento emulátor je jedním z nejnovějších na trhu. I přes svůj název jej lze používat bez nainstalování vývojového prostředí Visual Studio. Jako novinku přináší nové možnosti v konfiguraci, jako například nastavení místa pro geolokaci telefonu. Zároveň poskytuje vysoký výkon a stabilitu.

Problém tohoto emulátoru zatím je, že neobsahuje Google Play, a to způsobuje komplikace spojené s instalací aplikací na toto zařízení. (www.techadvisor.co.uk)

2.11 Možnosti tvorby Android Aplikací

V oblasti vývoje mobilních aplikací je více způsobů, jak k vývoji přistupovat. Jedním z rozhodnutí je volba vývojového prostředí, ve kterém aplikaci vyvíjet, dále pak volba emulátoru, který při vývoji využívat, ale za nejdůležitější rozhodnutí lze považovat způsob, jakým bude aplikace vyvíjena.

Před zahájením vývoje je potřeba zvážit možnosti, kterými lze mobilní aplikace vyvíjet. Způsobů tvorby aplikací je mnoho, proto je nutné vybírat jen z těch relevantních a aktuálně nejvyužívanějších.

Vybrané možnosti tvorby Android aplikací:

- Nativní aplikace – Java / Kotlin / C / C++,
- React Native,
- Webové HTML5 aplikace / Progressive Web Apps,
- Hybridní aplikace.

2.11.1 Nativní aplikace

Pro tvorbu aplikací pro operační systém Android se typicky využívá programovací jazyk Java. S tímto jazykem přišla společnost Sun Microsystems, Inc. roku 1995. Jednalo se o verzi 1.0 a způsobila velký pokrok v oblasti interaktivity webového prostředí. Postupem času se programovací jazyk vyvíjel a často přijímal nové funkce. (Schildt 2016)

Java se stala dominantním programovacím jazykem tvorby mobilních aplikací pro platformu Android, avšak v roce 2017 byl za oficiální jazyk určený pro tento účel vybrán programovací jazyk Kotlin.

Programovací jazyk Kotlin byl představen roku 2011 společností JetBrains. Kotlin lze využít pro psaní nejen mobilních aplikací, ale i webových aplikací, a to jak na straně klienta, tak na straně serveru. Patří mezi funkcionální jazyky a je zde vidět velká snaha o moderní přístup, příkladem mohou být datové třídy, které automaticky generují metody, jakými jsou `equals`, `hashCode`, `toString`, a další. Tím šetří čas programátora a udržují kód čitelnější a kratší. (www.xenonstack.com)

V případech, kdy je cílem vytvářet aplikace, u kterých je nesmírně důležitá rychlá odezva a vysoký výkon, je možné využít jazyky C a C++. Jak již bylo zmíněno, lze tyto programovací jazyky využít pomocí NDK, tedy nativního vývojářského souboru pro vývoj softwaru. (www.developer.android.com)

2.11.2 Webové HTML5 / Progressive Web Apps

HTML5 – Responsivní Webové Aplikace

Tyto aplikace je možno považovat za webovou stránku, případně více webových stránek určených k fungování na malé obrazovce mobilního zařízení. Velkou výhodou těchto aplikací je možnost přenesení znalostí vývoje webových stránek pomocí značkovacího jazyka HTML5.

Hlavní snahou těchto aplikací je vytvoření responsivního zobrazení pro mobilní zařízení, avšak problémem těchto aplikací je zejména u zařízení s operačním systémem Android, která často využívají různé prohlížeče, a tím může docházet k tomu, že budou zobrazovat aplikaci rozdílně.

Další nevýhodou je nemožnost využívání lokálního úložiště mobilního zařízení, obtížná garance bezpečnosti dat, a navíc zde není k dispozici tzv. offline režim, takže se aplikace může dostat do velmi nepředpokládaných stavů, případně se často vracet do stavu výchozího.

Zajímavostí těchto aplikací je, že jakožto webové aplikace je není možné umísťovat do prostředí Google Play, ani App Store. Tyto aplikace jsou pro uživatele dostupné přímo z webového prohlížeče, to v určitých případech může znamenat velkou výhodu.

Progressive Web Apps

Problémy zmíněné výše vedly společnost Google k práci na PWA, neboli Progressive Web Applications, které se snaží řešit zejména problémy offline režimu a ukládání dat. Zůstává zde však problém sníženého výkonu, jelikož aplikace běží v prostředí webového prohlížeče.

Za rok vzniku tohoto způsobu tvorby aplikací se považuje rok 2015. Od té doby má tento způsob tvorby aplikací velkou popularitu u vývojářů.

Problém, který přetrvává u těchto aplikací, je pak zejména nezájem společnosti Apple, který způsobuje, že tyto aplikace mají problémy se zařízeními, která fungují s operačním systémem Apple iOS. (www.simpleprogrammer.com)

2.11.3 Hybridní

Jak napovídá název, jedná se o kombinaci zmíněných HTML5 a nativních aplikací. Webová aplikace využívající HTML5, kaskádové styly a Javascript je obalena do nativního kontejneru. Tím lze využívat znalosti tvorby webových stránek bez omezení vyplývajících z HTML5 aplikací a s podporou přenositelnosti mezi platformami. Zásadním problémem je, že v porovnání s nativními aplikacemi jsou znatelně výkonnostně pomalejší. (www.developer.salesforce.com)

Cordova

Cordova je programový framework, který slouží k tvorbě hybridní mobilních aplikací. Vznikl okolo roku 2009 s názvem PhoneGap, který byl v roce 2012 přejmenován na Cordova. Tento framework slouží k tvorbě hybridních aplikací, které působí pro vývojáře webových aplikací intuitivním dojmem, z čehož vyplývá, že vývoj těchto aplikací je velmi rychlý. (www.makehybridapps.com)

2.11.4 React Native

Každá z možností má své výhody a nevýhody, avšak často je vidět velkou snahu o možnost využití znalostí tvorby webových aplikací při tvorbě aplikací mobilních. Příkladem tohoto přístupu je také framework React Native. React Native slouží pro tvorbu mobilních aplikací používáním knihovny React, jež byla vyvinuta společností Facebook. React Native vznikl v roce 2013 jako hackathonový projekt. Veřejnosti byl představen v lednu roku 2015. (www.code.facebook.com)

React Native využívá kód vytvořený v programovacím jazyce Javascript, který využívá Objective-C programové rozhraní, příp. Java programové rozhraní, pomocí kterých vykresluje nativní komponenty, na základě toho, zdali je aplikace určena pro iOS nebo Android. (www.discoversdk.com)

Toto je zcela zásadní rozdíl oproti hybridním aplikacím, jelikož zde výsledkem není HTML5 a Javascript zabalený do nativního kontejneru, ale výsledkem je zcela nativní programový kód. Z toho vyplývá také to, že programový kód napsaný s využitím programovacího jazyka Javascript lze využít pro vývoj aplikací jak pro Android, tak pro iOS, případně jakýkoliv jiný operační systém.

3 Analýza současného stavu

V této kapitole je představena společnost Tieto s.r.o., ve které se práce vykonávala. Také je zde detailně analyzován problém, který je zároveň v práci řešen.

Během analýzy budou nejprve definováni aktéři, základní funkcionality aplikace a jejich specifikace.

V rámci fáze analýzy bude při popisu struktury dat, se kterými bude aplikace pracovat, vytvořen doménový model.

Důležité je také provést analýzu budoucího užívání aplikace z pohledu uživatele, včetně komunikace mobilního zařízení se vzdáleným serverem. Z toho důvodu bude tento proces namodelován Diagramem aktivit.

Představení společnosti

Společnost Tieto s.r.o. zaměstnává v Ostravě přes 2000 lidí, od studentů až po naprosté špičky v oblasti informačních technologií. Společnost má sídlo v Helsinkách, pobočky pak zejména v České republice a Indii. V České republice je hlavní pobočka v Ostravě, další významnější je v Brně.

Popis problému

Společnost Tieto s.r.o. má mnoho pracovníků a klientů převážně z Finska a Indie. Společnost dbá na to, aby komunikace mezi pracovníky, případně mezi pracovníky a klienty nebyla pouze vzdálená, ale aby docházelo k přímému kontaktu.

Za tímto účelem je nutné organizovat velké množství návštěv zejména interních zaměstnanců firmy pracujících v jiných lokalitách do ostravské pobočky.

V současnosti je celková problematika návštěv řešena prostřednictvím elektronické pošty obsahující informace spojené s návštěvou a plánovaným programem.

Toto řešení je zastaralé a technicky nemoderní. Také díky rozsahu společnosti Tieto s.r.o. může docházet ke komplikacím spojeným s rozsahem a počtem osob, kterých se tato problematika týká.

Proběhla i snaha ze strany společnosti Tieto s.r.o. tento problém vyřešit vytvořením webové aplikace spolu s řešením zasílání dat o událostech ze strany serveru. Toto řešení bylo přijato pracovníky, kteří jsou zodpovědní za organizaci návštěv. Pro návštěvníka je však toto řešení nevhodné.

Hlavním problémem je, že webová aplikace není přizpůsobena pro používání na mobilních zařízeních a u většiny návštěvníků je nutné očekávat, že bude využívána téměř pouze na mobilních zařízeních s operačním systémem Android.

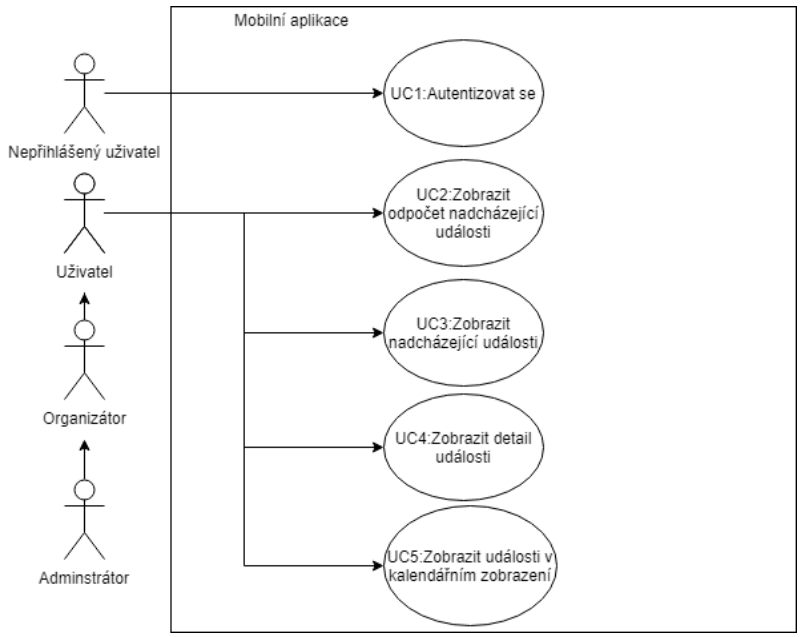
Jelikož se jedná o reprezentativní prvek společnosti, se kterým přijdou do kontaktu i klienti společnosti Tieto s.r.o., jeví se řešení spočívající v jednoduché optimalizaci webové aplikace jako nedostatečné.

3.1.1 Funkcionality aplikace

Jelikož se jedná o mobilní aplikaci určenou pro návštěvníky, bude naším cílem brát v potaz pouze funkcionality pro aplikaci zcela zásadní a eliminovat ty funkcionality, které by mohly negativním způsobem ovlivnit intuitivnost jejího používání.

V schématu 3.1 Diagramu užití jsou zobrazeni jednotliví aktéři, kteří mohou aplikaci užívat, a jednotlivé funkcionality očekáváné od aplikace.

Schéma 3.1 Diagram užití



Zdroj: vlastní zpracování

V schématu 3.1 je vidět, že i když je aplikace určena pro návštěvníky, mohou ji využívat i organizátoři, případně administrátoři. Administrátor představuje nejvyšší roli v aplikaci.

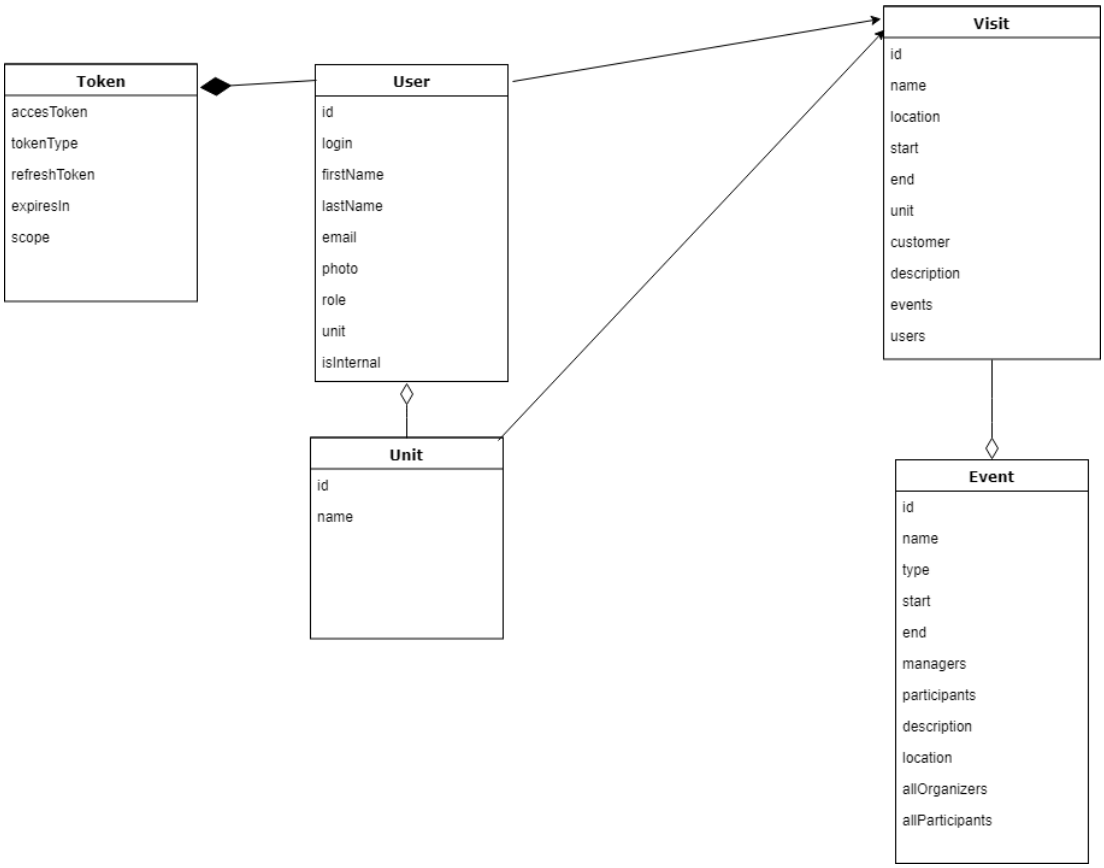
Na obrázku jsou zobrazeny i případy užití aplikace. Specifikace jednotlivých případů lze nalézt v příloze č. 1.

3.1.2 Struktura dat v aplikaci

Mobilní aplikace bude pracovat s daty uživatelů a jejich návštěv, proto je vhodné navrhnout, jaká bude struktura těchto dat.

K návrhu je využito doménového modelu, a to zejména proto, že není určen pro konkrétní programovací jazyk. Doménový diagram je zobrazen v schématu 3.2.

Schéma 3.2 Doménový diagram



Zdroj: vlastní zpracování

V schématu 3.2 je zobrazen objektově orientovaný návrh struktury dat v aplikaci. `User` reprezentuje uživatele aplikace. Mezi jeho atributy jsou obsaženy zejména základní informace o uživateli a reference na organizační jednotku, do které uživatel patří, proto je mezi nimi vztah agregace.

Autentizační token je ve schématu nazván `Token`. Slouží k potvrzení, že je uživatel autentizován, a obsahuje základní informace týkajících se typu tokenu a data jeho expirace. `Token` bez uživatele postrádá smysl, proto je mezi ním a uživatelem vztah kompozice.

`Unit` představuje organizační jednotku. Kromě identifikátoru má organizační jednotka jako svůj atribut název.

`Visit` reprezentuje ve schématu obecné informace o připraveném plánu návštěvy. Ve schématu je zobrazen také vztah s organizační jednotkou, která představuje oddělení zodpovědné za organizaci návštěvy. `Visit` také obsahuje reference na uživatele, pro které byla návštěva připravena a události, které obsahuje.

`Event` neboli událost je součástí `Visit`, avšak nemusí být součástí jen jedné. Proto je mezi entitami `Event` a `Visit` znázorněn vztah agregace. Události obsahují například informace o místě konání a základní popis události.

3.1.3 Komunikace se serverem

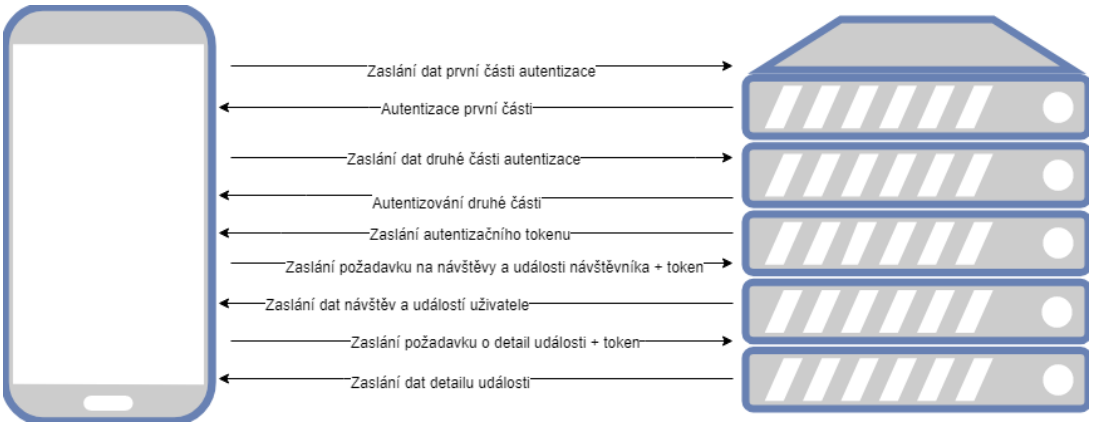
Za účelem návrhu komunikace mezi mobilní aplikací a serverem je vhodné modelovat, jakým způsobem bude probíhat užívání aplikace a kdy bude muset komunikovat se stranou serveru.

Pro modelování komunikace mezi mobilním zařízením a serverem byl využit softwarový program Bizagi (www.bizagi.com).

Schéma vytvořené prostřednictvím zmíněného programu je v příloze č. 2 (CD).

Pro nastínění obecného přehledu o komunikaci bylo vytvořeno i jednodušší schéma, zobrazené v schématu 3.3, které je soustředěno pouze na komunikaci mezi mobilním zařízením a serverem.

Schéma 3.3 Zjednodušený tok dat mezi mobilním zařízeném a serverem



Zdroj: vlastní zpracování

V schématu 3.3 je zobrazena zmíněná komunikace mezi mobilním zařízením a serverem. Nejprve je nutná autentizace uživatele. Po úspěšné autentizaci uživatele je do zařízení uložen autentizační token, který je s každým následujícím požadavkem zasílán, aby nebylo nutné opakování fázi autentizace při každém požadavku.

4 Návrh a realizace mobilní aplikace

Při návrhu je nutné rozhodnout o základních otázkách, jako je rozložení prvků v jednotlivých obrazovkách, aby bylo zajištěno intuitivní a přátelské prostředí pro uživatele. Také je nutné obrazovky vhodně uspořádat, aby navigace mezi nimi nepůsobila rušivým dojmem.

K tomuto základnímu návrhu je vhodné použít wireframes. Wireframes představují základní obrysy, které slouží k získání prvotní představy o návrhu.

Následně po návrhu aplikace je nutné zvolit i zmíněný způsob tvorby aplikace. Toto rozhodnutí je kritické, protože zvolení nesprávného způsobu může stát zbytečně čas, peníze a ve výsledku by daná aplikace nemusela splňovat očekávání. Je nutné zvážit aspekty související s jednotlivými způsoby tvorby mobilních aplikací pro platformu Android.

Po zvolení způsobu tvorby aplikace je možné začít se samotnou implementací. Nejprve je však nutné představit zvolený způsob tvorby a jeho základní pojmy, případně i balíčky, které mohou být při tvorbě aplikace využity.

Je také nutné zvolit minimální podporovanou verzi operačního systému Android.

Závěrem této části je shrnutí využitých technologií a práce s nimi. Součástí je také zhodnocení výsledků implementace aplikace, zhodnocení správnosti zvoleného způsobu tvorby, návrhy na vylepšení do budoucna a možnost distribuce finální verze aplikace.

4.1 Návrh aplikace

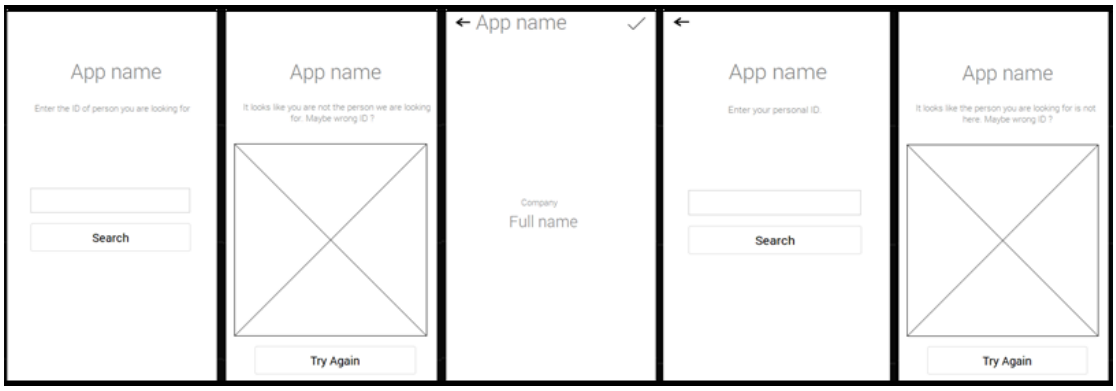
Uživatelské rozhraní je v mobilních, ale i webových aplikacích nesmírně důležité. Intuitivní a příjemný design je základem k úspěchu.

Aplikace je rozdělena do několika částí.

První částí je část autentizační, ve které musí dojít k ověření, že se skutečně jedná o osobu, pro kterou je návštěva připravena. Tato část nesmí být matoucí ani zbytečně komplikovaná, jelikož se jedná o první interakci uživatele s aplikací a negativní dojem by se těžko napravoval.

Tato část je zobrazena na obrázku 4.1. Jsou zde zobrazeny jednotlivé obrazovky autentizace, včetně případných chybových obrazovek.

Obrázek 4.1 Návrh autentizační části



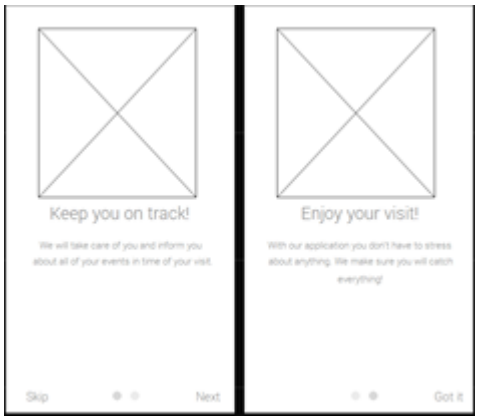
Zdroj: vlastní zpracování

Druhou, uvítací část, představují posuvné karty. Jedná se o mezičlánek mezi autentizační částí a hlavní částí aplikace. V této fázi je uživatel již ověřen a jedná se o přátelské přivítání do aplikace.

V této fázi je velká pravděpodobnost, že bude zařízení připojeno k internetu, proto je dalším účelem této části získání potřebných dat ze serveru. Zejména se jedná o informace o plánovaných událostech, které by byly po získání uloženy do paměti telefonu, a tím vznikne možnost, aby aplikace fungovala v offline režimu, tedy bez připojení k internetu.

Posunové karty jsou zobrazeny na obrázku 4.2.

Obrázek 4.2 Návrh uvítací části



Zdroj: vlastní zpracování

Třetí, hlavní část aplikace, tvoří dvě hlavní záložky.

Na první záložce uživatel uvidí nejbližší konanou událost, respektive její název a odpočet, jak dlouho zbývá do začátku. Rovněž zde budou zobrazeny i další probíhající a nejbližší události s jednoduchým popisem.

Na druhé záložce bude kalendářní zobrazení jednotlivých událostí. Uživatel bude mít tedy přehled i do budoucna o tom, co jej čeká, a stisknutím karty události může zjistit další

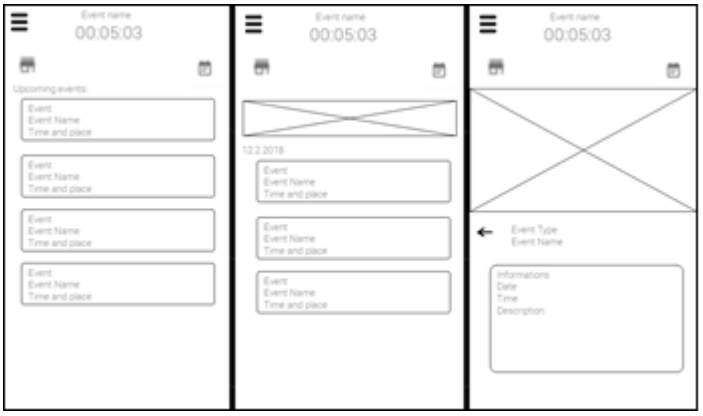
informace o události. Zde bude nutné kontrolovat, zdali je mobilní zařízení uživatele připojeno k internetu, aby bylo možné získat data ohledně zvolené události ze serveru.

V tomto případě je opět vhodné pokusit se o vytvoření možnosti fungování aplikace offline, tedy bez připojení k internetu, aby uživatel byl v případě, kdy není připojen, stále schopen využívat aplikaci, byť jen v omezené míře.

Jelikož jsou na první i druhé záložce zobrazeny události pouze s jednoduchým popisem, tak je nutné vytvořit obrazovku detailního zobrazení, na kterou se uživatel dostane stiskem prvku představující událost. Bylo by vhodné, aby v detailním zobrazení viděl návštěvník i mapu, kde bude zobrazeno místo konání události a veškeré informace související s událostí.

První a druhá záložka hlavní části jsou spolu s detailem události zobrazeny na obrázku 4.3.

Obrázek 4.3 Návrh hlavní části



Zdroj: vlastní zpracování

4.2 Volba způsobu tvorby aplikace

V této části práce bude řešena volba optimálního způsobu tvorby mobilní aplikace pro platformu Android.

Možností, jak vytvořit mobilní aplikaci je více a zvolení správné varianty je kritické. Při výběru špatné varianty může být zbytečně plýtván čas i peníze, případně může dojít i k tomu, že výsledná aplikace nebude splňovat zadané požadavky.

Cílem této kapitoly je tedy zvolení optimálního způsobu tvorby mobilní aplikace na základě předem stanovených požadavků a kritérií. Účelem bude získání ideálního způsobu tvorby Android aplikace.

V rámci volby nejlepší varianty budou zastoupeny 4 varianty, které budou hodnoceny na základě 4 kritérií. Variant i kritérií by mohlo být více, ale v této části jsou brány v potaz

pouze ta kritéria, která mají relevantní roli v rozhodování a varianty, které jsou aktuálně moderní a nejvíce využívané.

Je nutno zmínit, že variantami jsou míněny způsoby tvorby aplikací pro platformu Android, které byly představeny v kapitole 2. Zároveň není zahrnuto kritérium funkčnosti v offline režimu, jelikož každá z vybraných variant poskytuje tuto možnost.

4.2.1 Množina kritérií

Každé kritérium je označeno K a číslo kritéria. Toto značení bude využito i dále v textu. Jednotlivá čísla kritérií jsou značena vždy vedle názvu kritéria v nadpisech uvedených níže.

K1 – Výkon

Kritérium Výkon je v mobilních aplikacích velmi důležité, jelikož uživatel mobilních aplikací je v dnešní době již zvyklý na určitý standard a pokud by při běhu aplikace docházelo k jakýmkoliv rušivým elementům, jakými jsou pomalé načítání, dlouhá doba odezvy, případně animace, které nejsou plynulé, uživatel velmi rychle ztrácí zájem takovou aplikaci užívat.

Kritérium bude hodnoceno dle bodové škály od 1 do 5, která představuje možný výkon dané mobilní aplikace při využití daného způsobu. Popis jednotlivých bodových hodnot je zobrazen v tabulce 4.1. Body budou určovány zejména na základě způsobu, jakým mezi sebou zařízení a aplikace komunikují a zdali z toho vycházejí důsledky, které by mohly znepříjemnit uživatelský komfort při používání aplikace. Jedná se tedy o kvalitativní, ordinální typ dat, kdy jde o maximalizační funkci a vyšší hodnoty jsou preferovány před těmi nižšími.

Tabulka 4.1 Bodová škála výkonu

Výkon	Bodové ohodnocení
Vysoký	5
Střední – vysoký	4
Střední	3
Nízký – střední	2
Nízký	1

Zdroj: vlastní zpracování

K2 – Přenositelnost mezi platformami

V dnešní době je nutné se dívat do budoucna. S tím souvisí i přenositelnost mezi platformami. Aplikace je implementována primárně pro platformu Android, ale každý

zadavatel by ocenil možnost využití aplikace i na jiných platformách, zejména na iOS, protože jinak by byl nucen v případě úspěchu této aplikace a nemožnosti přenést aplikaci na jinou platformu vytvořit novou aplikaci od začátku, což stojí čas a peníze.

I toto kritérium je hodnoceno škálou od 1 do 5, kdy 1 znamená, že přenos je prakticky nemožný a 5 znamená, že přenos je bezproblémový. Bodové hodnocení je zobrazeno v tabulce 4.2. Jedná se opět o kvalitativní, ordinální typ dat a maximalizační funkci, ve které je snaha získat co nejvyšší hodnoty.

Tabulka 4.2 Bodová škála přenositelnosti

Přenos mezi platformami	Bodové ohodnocení
Bezproblémový	5
S malými problémy	4
Středně problémový	3
Velmi problémový	2
Téměř nemožný	1

Zdroj: vlastní zpracování

K3 – Rychlost vývoje

Rychlost vývoje je v dnešní době, kdy cena hodiny práce programátora stále roste, pro každého zadavatele projektu důležitá.

Toto kritérium je těžce měřitelné, jelikož v realitě jde o záležitost subjektivní. Přesto je to velmi důležité kritérium při rozhodování. Proto je toto kritérium posuzováno na základě obecných informací týkajících se rychlosti vývoje.

Kritérium je hodnoceno na škále od nejpomalejšího vývoje číslem 1, až po nejrychlejší označené číslem 5. Cílem je maximalizovat tuto hodnotu a jedná se o kvalitativní, ordinální typ dat. Bodová škála je zobrazena v tabulce 4.3.

Tabulka 4.3 Bodová škála rychlosti vývoje

Rychlost vývoje	Bodové ohodnocení
Velmi rychlá	5
Rychlá	4
Středně rychlá	3
Pomalá	2
Velmi pomalá	1

Zdroj: vlastní zpracování

K4 – Distribuce

Způsob distribuce aplikace koncovým uživatelům je velmi důležitá součást rozhodování. Jsou zde v zásadě dvě varianty, a to distribuce prostřednictvím webového prohlížeče, nebo prostřednictvím Google Play.

Typ dat je zde kvalitativní, dichotomický a obsahuje dvě hodnoty: prohlížeč, nebo Google Play. V tomto případě preferujeme hodnotu Google Play před prohlížečem, protože se jedná o interní aplikaci a distribuce se bude lépe kontrolovat díky možnosti vydávat privátní aplikace. Google Play tedy představuje hodnotu 1 a prohlížeč hodnotu 0 s tím, že cílem je danou hodnotu maximalizovat.

4.2.2 Množina variant

Každá varianta je označena písmenem V následovaným číslem varianty. Toto značení bude využito i v následující části této analýzy. Varianty vycházejí z teoretické části, kde byly představeny a popsány. Pro přehlednost jsou zobrazeny v tabulce 4.4.

Tabulka 4.4

Značení varianty	Název varianty
V1	Nativní – Java, Kotlin, C, C++
V2	React Native
V3	Hybrid
V4	Webové HTML5 / Progressive Web Apps

Zdroj: vlastní zpracování

4.2.3 Stanovení vah kritérií

Pro zjištění optimální varianty je třeba si uvědomit, jak je které kritérium důležité. To bude představovat vypočtená váha každého kritéria. V analýze bude pro výpočet vah kritérií využita Saatyho metoda.

Saatyho metoda

Saatyho metoda je jednou z komplexnějších metod pro zjištění vah jednotlivých kritérií. Nevýhodou této metody je nutnost provádět komplikovanější výpočet oproti ostatním metodám, ale na druhou stranu je výhodou využití této metody větší přesnost výsledků a možnost jejich ověření.

V prvním kroku této metody jsou porovnávány kritéria mezi sebou. V této metodě jsou využity hodnoty preferencí, které vycházejí ze Saatyho škály zobrazené v tabulce 4.5.

Tabulka 4.5 Saatyho škála

Slovní ohodnocení	Číselná hodnota
Extrémně důležité	9
	8
Velmi silně více důležité	7
	6
Silně více důležitě	5
	4
Střídmě více důležitě	3
	2
Stejně důležité	1

Zdroj: (Mu, Pereyra-Rojas, 2016) – vlastní zpracování

V matici platí, že všechny diagonální prvky mají hodnotu 1 a pro všechny prvky platí vztah vyjádřený dle vzorce 4.1. S značí Saatyho matici.

$$s_{ij} = \frac{1}{s_{ji}} \tag{4.1}$$

Výpočet vah, značených w, kritérií se vypočítá pomocí vzorce 4.2, za podmínky $\sum_{j=1}^{n} w_j = 1$. Váha kritéria je tedy vypočtena jako geometrický průměr řádku matice vůči sumě geometrických průměru všech řádků.

$$w_i = \frac{\left[\prod_{j=1}^{n} s_{ij}\right]^{\frac{1}{n}}}{\sum_{k=1}^{n}\left[\prod_{j=1}^{n} s_{ij}\right]^{\frac{1}{n}}} \tag{4.2}$$

V tabulce 4.6 jsou zobrazena jednotlivá kritéria a jejich porovnání mezi sebou. Jsou zde zobrazeny zmíněné preference.

Tabulka 4.6 Saatyho matice

Kritérium	K1	K2	K3	K4
K1	1,00	2,00	5,00	4,00
K2	0,50	1,00	4,00	3,00
K3	0,20	0,25	1,00	3,00
K4	0,25	0,33	0,33	1,00
Konzistence matice				9,5 %

Zdroj: vlastní zpracování

V tabulce 4.7 jsou zobrazeny výsledky geometrického průměru, ze kterého vycházejí výsledné hodnoty vah, které jsou následně pro přehlednost vynásobeny 100 a tím vyjádřeny v procentech.

Tabulka 4.7 Výsledky Saatyho metody

Kritérium	Geometrický průměr	Výsledné váhy	Výsledné váhy v %
K1	2,51	0,49	49 %
K2	1,57	0,31	31 %
K3	0,62	0,12	12 %
K4	0,41	0,08	8 %

Zdroj: vlastní zpracování

Jak bylo zmíněno, lze u Saatyho matice výpočtem zjistit její konzistentnost. V případě, že by poměr konzistence vycházel vyšší než 10 %, bylo by na našem zvážení přepočítat matici, ze které vycházejí příslušné výsledky, případně se zamyslet nad jejími hodnotami. V tabulce 4.6 je zobrazena hodnota konzistence matice 9,5 %, která je menší než zmíněných 10 %, což tedy znamená, že matice je konzistentní.

Výpočet se provádí pomocí vzorce $CR = \frac{CI}{RI}$, kde CR je zmíněný poměr konzistence, RI je náhodný index, který vychází z tabulky 4.8 a CI je index konzistence, který se vypočítá pomocí vzorce $CI = \frac{\lambda_{max} - k}{k-1}$, kde k značí počet kritérií a λ_{max} je největší vlastní číslo matice. (Mu, Pereyra-Rojas, 2016)

Tabulka 4.8 Náhodný index

K	1	2	3	4	5	6	7	8	9	10
Náhodný index	0	0	0,58	0,9	1,12	1,24	1,32	1,41	1,45	1,49

Zdroj: (Saaty, 1987) – vlastní zpracování

Z výsledků zobrazených v tabulce 4.7 je patrné, že nejdůležitější kritérium je výkon, poté následuje přenositelnost. Méně důležitým kritériem se jeví rychlost vývoje a distribuce.

4.2.4 Aplikace metody rozhodování

V této kapitole bude využita metoda PROMETHEE. (www.promethee-gaia.com). Metoda bude řešena s využitím softwarových nástrojů.

Metoda PROMETHEE

Metoda PROMETHEE patří mezi metody vícekriteriálního rozhodování. Tato metoda je založena na preferenčních relacích, které vychází z párového porovnání jednotlivých

variant. Jejím využitím dojde k určení zmíněných vztahů preferencí, pomocí kterých bude zvolena optimální varianta způsobu tvorby aplikace.

Při použití metody PROMETHEE patří mezi její vstupy zejména množina kritérií a množina variant. Také ohodnocení každé varianty z pohledu kritéria a relace, které vyjadřují preference pro každé kritérium a váhy kritérií.

Mezi výstupy je očekáváno zejména ohodnocení variant a jejich vzájemná porovnatelnost. Také je očekáváno zjištění slabých a silných stránek jednotlivých variant.

Metoda vychází z výpočtu kladných a záporných toků. Porovnává se, jak si varianta vede s jinými na základě jednotlivých kritérií.

Pro výpočet kladných toků se využívá vzorec 4.3, kde pozitivní tok $\Phi^{+}(a)$ udává, jak je preferována daná varianta vůči všem ostatním, n značí počet variant a $\pi(a,b)$ značí index preference, který vyjadřuje vážený preferenční stupeň při porovnání dvou variant dle daného kritéria. Index preference je v intervalu $\pi(a,b) \in [0,1]$, kde 0 znamená nepreferování a vůči b a 1 znamená silnou preferenci a vůči b.

$$\Phi^{+}(a) = \frac{1}{n-1}\sum_{b \neq a} \pi(a,b) \tag{4.3}$$

Pro výpočet záporných toků se využívá vzorec 4.4, kde záporný tok $\Phi^{-}(a)$ udává, jak moc jsou preferovány ostatní varianty vůči dané variantě a n značí počet variant. Index preference je při výpočtu záporných toků stejný jako v případě výpočtu kladných toků, avšak nevyjadřuje preference a vůči b, ale b vůči a.

$$\Phi^{-}(a) = \frac{1}{n-1}\sum_{b \neq a} \pi(b,a) \tag{4.4}$$

Když se záporné toky odečtou od toků kladných, dojde k získání hodnoty čistého preferenčního toku. Varianta s nejvyšší hodnotou je variantou nejlepší.

Výběr optimální varianty

Pro metodu PROMETHEE byl využit softwarový program Visual PROMETHEE, který jednak ulehčuje výpočty a poskytuje i mnoho zajímavých grafických zobrazení výsledků, včetně nástrojů pro citlivostní analýzu, které však nejsou v práci využity. Vstupy pro výpočty jsou zobrazeny v tabulce 4.9.

Tabulka 4.9 Vstupní data

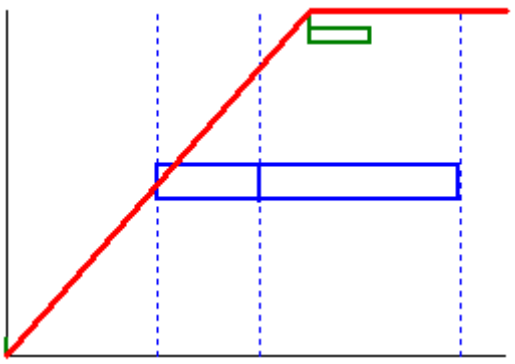
Criteria	Výkon	Přenositelnost	Rychlost výv...	Distribuce
Unit	5-point	5-point	5-point	store/web
Cluster/Group				
Preferences				
Min/Max	max	max	max	max
Weight	49,00	31,00	12,00	8,00
Preference Fn.	Linear	Linear	Linear	Usual
Thresholds	absolute	absolute	absolute	absolute
- Q: Indifference	0	0	1	n/a
- P: Preference	2	3	3	n/a
- S: Gaussian	n/a	n/a	n/a	n/a
Statistics				
Minimum	2	1	2	0,0
Maximum	5	5	5	1,0
Average	4	3	4	0,8
Standard Dev.	1	2	1	0,4
Evaluations				
Java/Kotlin/C++/C	5	1	2	store
React Native	4	4	3	store
Hybrid	2	5	4	store
Web/PWA	3	2	5	web

Zdroj: vlastní zpracování

Pro kritérium distribuce, které je dichotomické, je vhodné využít funkci Usual, neboli obvyklou, ale u ostatních je vhodnější využít funkci Linear, tedy lineární. Lineární funkce dovoluje nastavit hodnotu indiference a preference. (www.promethee-gaia.com)

V grafech 4.1, 4.2 a 4.3 jsou zobrazeny jednotlivé preferenční funkce variant, které jsou lineární. Následně v grafu 4.4 je zobrazena preferenční funkce distribuce, která je Usual.

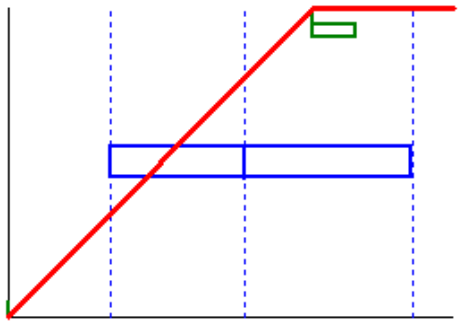
Graf 4.1 Preferenční funkce výkonu



Zdroj: vlastní zpracování

V grafu 4.1 je vidět, že jakýkoliv rozdíl v hodnotách hraje roli, jelikož hodnota indiference je 0. Také je zvolena preference 2, takže preference lepší varianty před horší lineárně roste, dokud je rozdíl mezi hodnotami menší než 2, ale s rozdílem rovnajícím se a vyšším než 2 je již jednoznačně preferována lepší varianta před horší.

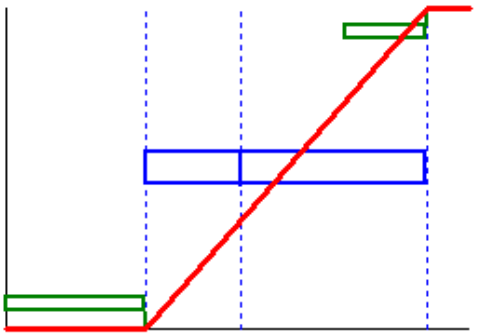
Graf 4.2 Preferenční funkce přenositelnosti



Zdroj: vlastní zpracování

Preferenční funkce přenositelnosti je obdobná, jako v případě preferenční funkce výkonu, avšak preference má hodnotu 3.

Graf 4.3 Preferenční funkce rychlosti vývoje

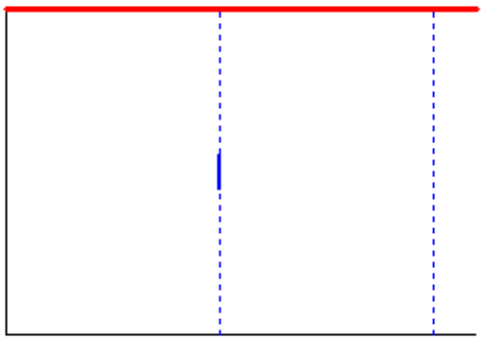


Zdroj: vlastní zpracování

V případě preferenční funkce rychlosti vývoje, zobrazené v grafu 4.4, je opět zvolena preference s hodnotou 3, ale je zde i indiference o hodnotě 1. To znamená, že rozdíl o 1 je indiferentní, takový rozdíl tedy nehraje roli, pokud má tedy jedna varianta hodnotu 5 a druhá hodnotu 4, tak v tom ve výsledku není rozdíl.

Indiference vychází zejména z toho, že kritérium rychlosti vývoje je velmi těžko hodnotitelné a pro získání co nejkvalitnějších výsledků je vhodné toto kritérium tímto upravit.

Graf 4.4 Preferenční funkce distribuce



Zdroj: vlastní zpracování

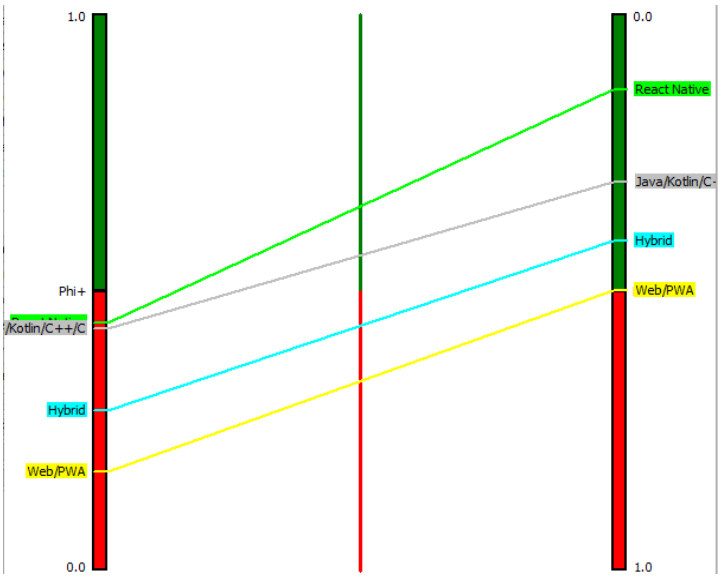
V grafu 4.4 je zobrazena preferenční funkce distribuce. Je patrné, že při jakémkoliv rozdílu hodnot je lepší varianta jednoznačně preferována před tou horší. V případě, že máme pouze dvě možnosti, je využití této preferenční funkce vhodné. Je důležité zmínit, že funkční hodnota preferenční funkce v bodě [0;0] je 0, jelikož to z obrázku nemusí být patrné.

Výsledky rozhodování

Na základě vložených vstupů vytvoří program Visual PROMETHEE mnoho výstupů. Pro přehlednost jsou v práci využity pouze nejzákladnější a vhodné k prezentaci výsledků.

Jako první a pro získání výsledků jeden z nejdůležitějších je Partial Rankings. Tento výstup graficky zobrazuje jednotlivé varianty a jejich kladné, záporné toky. Jak je vidět v grafu 4.5, jako nejlepší varianta vychází React Native, poté Java/Kotlin/C/C++, následuje Hybrid a poslední je Web/PWA. Zároveň se jednotlivé přímky nekříží, což znamená, že všechny varianty jsou vzájemně porovnatelné.

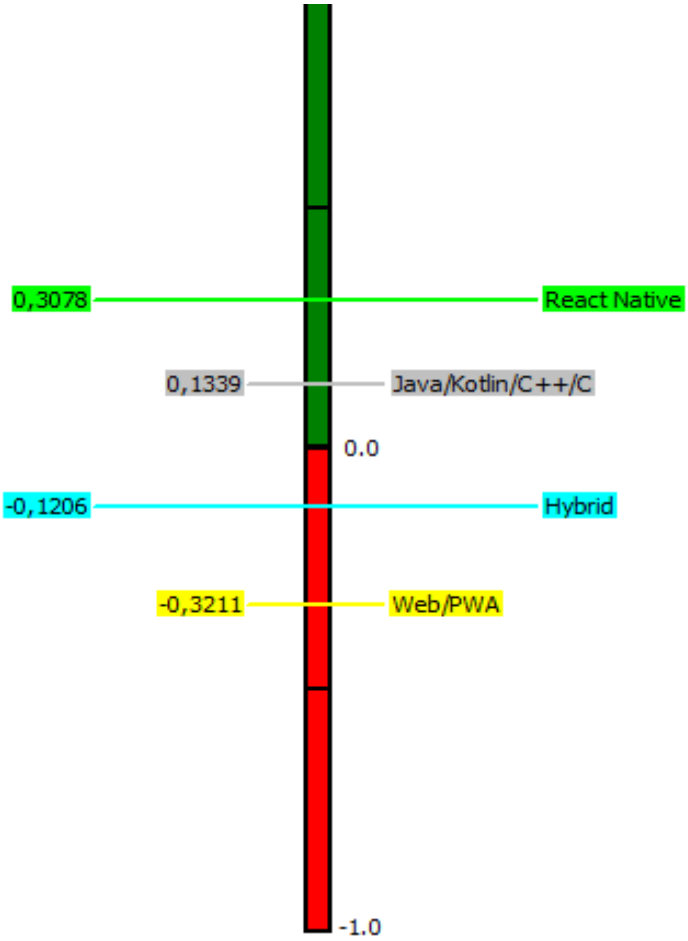
Graf 4.5 Partial Rankings



Zdroj: vlastní zpracování

Další zobrazení, které je znázorněno v grafu 4.6, představuje Complete Ranking, tedy celkové pořadí. Rozdíl oproti přechozímu je ten, že zde je vidět i hodnoty čistých preferenčních toků.

Graf 4.6 Complete Rankings



Zdroj: vlastní zpracování

V tabulce 4.10 jsou zobrazeny veškeré kladné i záporné toky. Jak je vidět v tabulce 4.10, tak vítěz React Native má oproti ostatním největší kladné toky a zároveň nejnižší toky záporné.

Tabulka 4.10 Tabulkové zobrazení

Rank	Variant	Phi	Phi+	Phi-
1	React Native	0,3078	0,4439	0,1361
2	Java/Kotlin/C++/C	0,1339	0,4350	0,3011
3	Hybrid	-0,1206	0,2878	0,4083
4	Web/PWA	-0,3211	0,1761	0,4972

Zdroj: vlastní zpracování

4.2.5 Výsledky volby optimální varianty

V rámci této části byla popsána kritéria. Těmto kritériím byla na základě analýzy přidělená váha důležitosti při rozhodování. Nejdůležitějším kritériem je tedy výkon, následované postupně přenositelností, rychlostí vývoje a distribucí.

Poté byly představeny varianty způsobu vývoje Android aplikací. Následná analýza pomocí metody PROMETHEE zvolila z variant Java/Kotlin/C/C++, React Native, Hybrid, Web/PWA jako optimální variantu React Native, která získala zejména kvůli vysokému výkonu a přenositelnosti mezi platformami nejlepší ohodnocení.

Na základě této analýzy došlo tedy k rozhodnutí využít technologii React Native pro implementaci aplikace.

4.3 Volba vývojového prostředí a emulátoru

Volbu vývojového prostředí a emulátoru již není nutno řešit analýzou, jelikož volba těchto pomocných nástrojů je subjektivní, pokud tomu nebrání okolnosti. V případě této práce nejsou stanovena žádná omezení ani podmínky zadavatele, proto volba vychází zejména z informací obsažených v druhé kapitole této práce.

Vývojové prostředí

Jako vývojové prostředí bylo zvoleno Android studio, jelikož se jedná o vývojové prostředí velmi kvalitní a má největší podporu pro vývoj aplikací pro platformu Android.

Toto rozhodnutí je zapříčiněno z části i tím, že využití vývojového prostředí WebStorm, jakožto nejlepší varianty pro tvorbu aplikace v jazyce Javascript, má poměrně náročné podmínky pro využití v komerčních aplikacích.

Pokud by zde nebyla tato komplikace, pravděpodobně by došlo ke zvolení vývojového prostředí WebStorm, zejména na základě popisu v druhé kapitole této práce a vhodnosti využití s frameworkem React Native.

Emulátor

Jako vhodný emulátor se nejprve zdál emulátor Genymotion. Popisovaný skvělý výkon a více možností oproti konkurenci z něj vytvořily vhodného kandidáta. Problém však nastal v tom, že oproti emulátoru Android studia působil velmi zpomaleným dojmem.

Tato záležitost by se pravděpodobně dala vyřešit, ale při práci se zmíněným emulátorem Android studia došlo k rozhodnutí, že splňuje veškeré požadavky na výkon a možnosti využití, a proto byl zvolen jako testovací emulátor.

Navíc je tento emulátor součástí vývojového prostředí Android studia, a proto je jejich vzájemná kooperace na vysoké úrovni a není nutné platit za využívání emulátoru Genymotion.

4.4 Základní prvky tvorby aplikací v React Native

Jako optimální varianta tvorby mobilní aplikace byla zvolena varianta tvorby pomocí React Native.

Před zahájením implementace je nutné tento způsob popsat podrobněji, než tomu bylo v kapitole 2. Důraz je kladen zejména na popis základních pojmů nutných k implementaci aplikace.

Také je nutné zmínit základní problémy, které vycházejí z tvorby aplikací tímto způsobem a balíčky, které tyto problémy řeší.

4.4.1 Komponenty

Deklarativní komponenty jsou prostředek k tvorbě bohatého uživatelského rozhraní. (www.facebook.github.io)

Komponenty se velmi často vnořují, pokud je potřeba předat z vyšší komponenty nižší nějaký parametr, tak je možné pomocí předání props, neboli vlastností.

4.4.2 JSX

JSX je zkratka pro JavaScript XML a je to preprocessor, pomocí kterého je možné vložit XML syntaxi do JavaScriptu. Pomocí JSX lze využívat komponenty Reactu, například <Text> je komponent pro zobrazení textu.

4.4.3 Stav

Každá komponenta může udržovat nějaký stav. Pokud jej udržuje, jedná se o chytrou komponentu. Takováto komponenta má konstruktor, ve kterém udržuje svůj stav.

Dobrým příkladem takové komponenty je nejvýše postavená komponenta, která bude mít pod sebou hierarchicky komponenty prezentační. Prezentační komponenty svůj stav neudržují, pouze přijímají props, které mohou využívat ve své vykreslovací funkci.

Uchovávání stavu v aplikaci je mnohdy velmi komplikované. Pokud na sobě jednotlivé prvky aplikace závisejí, případně se ovlivňují, dochází často k nechtěným následkům, což může znamenat ztrátu funkčnosti aplikace, nepředpokládané chování, vizuální chyby.

Řešení naskýtá balíček Redux. Jedná se o stavový kontejner, který pro zabezpečení předpokládaného chování aplikace využívá čistých funkcí. (www.redux.js.org)

Balíček Redux ovšem neřeší vedlejší efekty, jako jsou například asynchronní akce. Tuto záležitost řeší balíček Redux Saga, který lze snadno implementovat s balíčkem Redux. (www.redux-saga.js.org)

4.4.4 Navigace v rámci aplikace

Navigace v mobilních aplikacích musí být svižná, příjemně animovaná, ale zároveň pro programátora přehledná. Možností v této oblasti je mnoho, avšak oficiální řešení v této problematice ze strany společnosti Facebook není. (www. facebook.github.io)

React-router-navigation je balíček, který není jedním z vypsaných doporučených způsobů navigace v dokumentaci, ale vychází z balíčku React router, což je nejpoužívanější balíček pro řešením navigace ve webových aplikacích, využívajících knihovnu React. (www. github.com)

4.4.5 Ukládání dat do zařízení

V React Native se pro ukládání dat do paměti telefonu využívá asynchronní úložiště. Jedná se o jednoduchý, nezašifrovaný, asynchronní, persistentní úložný systém, který funguje na principu ukládání klíčů a hodnot.

Na platformě Android využívá asynchronní úložiště RocksDB nebo SQLite, na základě toho, co se na daném mobilním zařízení nachází. (www.facebook.github.io)

4.4.6 Offline režim

Mobilní aplikace často čelí problému, že zařízení, na kterých aplikace běží, nejsou připojena k internetu. Mobilní zařízení se může za běhu aplikace připojovat a odpojovat k internetu a tím způsobit nemalé problémy, pokud s tím aplikace nepočítá.

Aplikace musí být schopna adekvátně reagovat na změny stavu připojení. React Native poskytuje pro detekci připojení k internetu NetInfo, což je rozhraní, které umožňuji zjistit, zdali je uživatel připojen k internetu a případně i jakým typem připojení. (www.facebook.github.io)

4.4.7 Zobrazení míst na mapě

V mnohých aplikacích dochází k práci s geografickými údaji a může být vhodné je uživateli zobrazit na obrazovce zařízení. K tomu je možné využít balíček React-native-maps, který poskytuje komponenty, které po zadání zeměpisné šířky a zeměpisné délky vykreslí na obrazovce mapu. Mapě je možné nastavit měřítko a případně ji stylizovat dle potřeb vývojáře.

4.4.8 Časové údaje

Čas, jeho ukládání, zobrazování a další manipulace bývá ve světě informačních technologií náročným problémem. Zejména pokud se musí dbát i na různá časová pásma, protože pak může často docházet k nesrovnalostem.

Balíček Moment slouží k parsování, formátování časových údajů, včetně možnosti lokalizace. Dokáže také spočítat dobu mezi daty ve zvoleném formátu. (www.momentjs.com)

4.4.9 Minimální verze Java programového rozhraní

Jako minimální verze Java programového rozhraní je zvolena verze 16. Toto rozhodnutí vychází z oficiálních stránek React Native a je velmi příznivé, jelikož tato verze nepatří k těm nejnovějším, aplikace bude tedy dostupná pro velké množství zařízení. (www.github.com)

4.5 Řešení jednotlivých obrazovek

V této části bakalářské práce budou řešeny jednotlivé části aplikace. Všechny obrázky využité v aplikaci byly vytvořeny jako vektorové, prostřednictvím softwarové aplikace Adobe XD. (www.adobe.com)

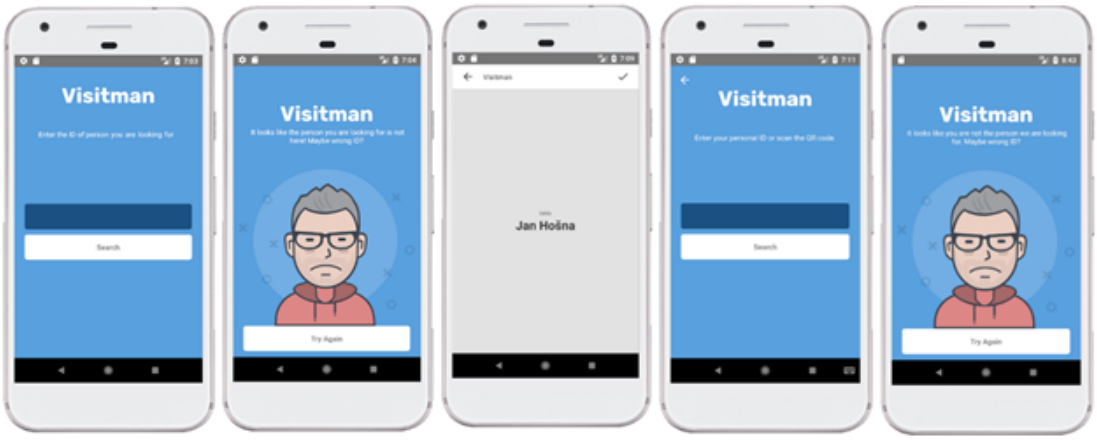
4.5.1 Autentizační část

Je nutné zvolit vhodný způsob autentizace, tedy ověření, že ten, kdo chce využívat aplikaci, je skutečně určená návštěva.

Možností je zde několik, každé má svá pro a proti. Vždy se jedná o určitý způsob kombinace uživatelského jména a hesla, ať už ve formě emailu a hesla, případně veřejného a privátního klíče, nebo existuje i možnost využití QR kódu, obsahující jistou ekvivalentní formu uživatelského jména a hesla.

Po zvážení byla zvolena varianta privátního a veřejného klíče. Tato varianta odpovídá návrhu a není nutné zasílat citlivé údaje. Na obrázku 4.4 jsou zobrazeny implementace jednotlivých obrazovek. Vycházejí z návrhu v předešlé části této kapitoly. Jako i v následujících obrázcích došlo k záměně některých obecných předpokladů.

Obrázek 4.4 Implementace autentizační části



Zdroj: vlastní zpracování

4.5.2 Úvodní část

V této části je uživatel již přihlášen a je přivítán první obrazovkou z úvodní části. Zároveň zde dochází k odeslání požadavků na server pro získání dat událostí určených pro návštěvníka. Tato část je řešena pomocí React-router-navigation `Cards`, které umožňují vytvořit routy neboli cesty, na jednotlivé karty.

Na obrázku 4.5 je zobrazena implementace úvodní části. Jelikož je v této části vhodné uživatele pozdržet, bylo vhodné vytvořit obrázek a text, který by zaujal pozornost uživatele, než budou získána data ze serveru.

Obrázek 4.5 Implementace úvodní části

Zdroj: vlastní zpracování

4.5.3 Hlavní část

Tato část představuje hlavní část aplikace, která se skládá ze dvou základních záložek. Tyto záložky jsou řešeny prostřednictvím balíčku React-router-navigation.

První záložka

V první záložce je v horní části zobrazen čas zbývající do nadcházející události a její název. Stiskem dojde k přesměrování na detailní obrazovku události. Pokud není žádná nadcházející událost, bude zobrazen nápis oznamující tuto informaci.

Tato část je řešena pomocí samostatné komponenty. Problémy související s časovými údaji jsou řešeny pomocí balíčku Moment.

Další částí této záložky jsou nejbližší události se základními informacemi o nich. Každá z těchto událostí je zobrazena na vlastní kartě. Jedná se opět o vlastní komponentu, která se využívá v rámci aplikace.

Druhá záložka

V druhé záložce jsou kalendářním způsobem zobrazeny jednotlivé události. Tvorba vlastního kalendáře zabrala mnoho času a výsledek byl zpočátku vizuálně v pořádku, ale při tahu prstem docházelo k vizuálním chybám a nedostatečně rychlé reakci na dotyková gesta. Dalším problémem byla dlouhá doba inicializace kalendáře. Pokud by uživatel chtěl přejít z první záložky na druhou, docházelo po stisku k několikavteřinovému čekání. To je samozřejmě nepřijatelné a bylo nutné najít jiné řešení.

Po čase stráveném studováním dokumentace a snaze o implementaci vlastního kalendáře se naskytlo nejlepší řešení, a to využití balíčku od společnosti Wix, React-native-calendars.

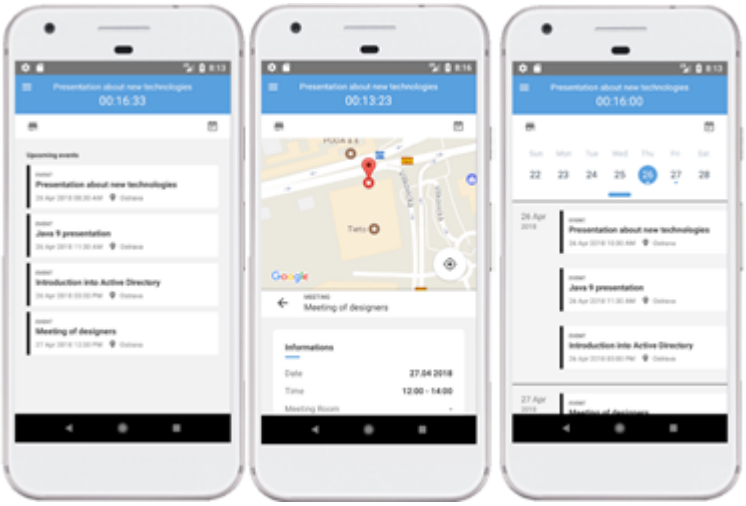
Tento balíček poskytuje vysoce optimalizovanou kalendářní komponentu, která se dá výrazně upravovat. Hlavní výhodou je velmi nízká odezva a dopracované animace přechodů.

Detail události

Každá karta, představující událost, se dá stisknout. Stisknutím karty dojde k zobrazení detailu události. V detailu jsou obsaženy základní informace obohacené o zobrazení místa konání události na mapě a další informace související s událostí, které by se do základní karty nevešly, jedná se zejména o popis události.

Implementace hlavní části je zobrazena na obrázku 4.6.

Obrázek 4.6 Implementace hlavní části



Zdroj: vlastní zpracování

4.6 Zhodnocení implementace a distribuce

Výsledkem předchozích kapitol je aplikace vytvořená pomocí frameworku React Native. Aplikace odpovídá návrhu a je připravena na praktické testování a následné uvedení do provozu. Jako nejtěžší část implementace se jevilo získání informací o možnostech využití různých balíčků a jejich využití.

Největší komplikace nastávají ve snaze o jejich kombinaci, jelikož se mnohdy jedná o balíčky, o které se starají vývojáři ve volném čase. Avšak samotný React Native často přechází na nové verze a tím vznikají problémy s kompatibilitou. Řešení většinou trvá dlouhou dobu, jelikož neexistuje jasný návod jak řešit tyto problémy a často je těžké identifikovat, čím jsou způsobeny. Proto je důležité využívat v aplikaci balíčky, které jsou udržované a nemají evidováno mnoho problémů spojených s kompatibilitou.

Distribuce aplikace proběhne s největší pravděpodobností jako soukromá aplikace distribuovaná přes Google Play. V případě velké firmy, jakou je Tieto s.r.o. není možné předpokládat okamžité využití aplikace. Oblast, kterou tato aplikace řeší, je poměrně velká a nejprve se s mobilní, tak i webovou aplikací musí naučit pracovat zaměstnanci mateřské firmy, kteří jsou zodpovědní za organizaci návštěv do této společnosti.

Detaily ohledně distribuce budou řešeny až po dostatečném otestování aplikace a rozhodnutí vedení o uvedení aplikace do provozu.

4.7 Návrhy na vylepšení aplikace

Aplikace ve verzi odpovídající návrhu je již implementována, avšak během fáze implementace se naskytlo mnoho nápadů, které by mohly aplikaci vylepšit a udělat pro uživatele zajímavější.

Na již zvolených technologiích se nic nemění, jelikož se osvědčily a není důvod je nahrazovat jinými. Změny se týkají zejména přidání čtečky QR kódů pro autentizaci, přidání třetí záložky v hlavní části pro zobrazení kategorií míst, zobrazení karet míst v první záložce hlavní části a přidání zobrazení detailu místa, včetně možnosti místa hodnotit.

Mezi oblasti zlepšení patří také řešení aktualizace dat v aplikaci, jelikož v její aktuální verzi dojde po přihlášení uživatele k získání dat, s kterými se po celou dobu běhu aplikace pracuje.

Aktualizace dat v aplikaci

Jako možné řešení se naskytuje využití vzdálených upozornění, avšak mohlo by dojít k tomu, že by uživatel nebyl připojen k internetu a k následným komplikacím.

Dalším řešením je aktualizace dat ve zvolených časových intervalech, ale i v tomto případě se může vyskytnout problém související s nepřipojením zařízení k internetu. Toto řešení je také nevhodné z důvodu, že připojení mobilních zařízení k internetu je často omezené, případně zpoplatněné, a časté aktualizace dat v aplikaci jsou tedy nepřijatelné.

Mezi další řešení patří možnost zasílání SMS zpráv, které by upozornily návštěvníka na změnu v programu.

Z řešení popsaných výše však žádné není zcela ideální a aktuální řešení je pro testovací účely dostatečné, avšak v dalších verzích bude nutné nahradit stávající řešení některou z uvedených variant, případně jejich kombinací.

Čtečka QR kódů

Autentizační část je již v provozu, ale přidání možnosti autentizace pomocí QR kódů je i pro uživatele velmi zajímavá. Jedná se o zpestření pro uživatele, ale i ulehčení jeho námahy oproti nutnosti zadávání klíčů.

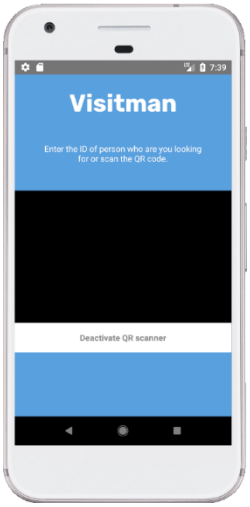
Pro implementaci bude využit balíček React-native-qrcode-scanner. Tento balíček je postaven na základu React-native-camera a poskytuje možnost naskenovat QR kód, ze kterého pak získá data. Tato data mohou představovat privátní klíč, který se po naskenování odešle jako v případě zadání zmíněného klíče.

Řešení je ve výsledku uživatelsky přívětivější a intuitivnější. Pokud by uživatel nechtěl QR kód využívat, případně by měl problémy s kamerou, zůstává stále možnost využití původního způsobu autentizace, tedy veřejným a privátním klíčem.

Na obrázku 4.7 je zobrazena implementace čtečky QR kódů v první části autentizace. Uživatel má nyní možnost aktivovat, případně deaktivovat tuto možnost.

Tento balíček má však problémy při využití na emulátoru. Na reálném zařízení bezproblémově funguje, avšak v emulátoru je místo zobrazení kamery černá oblast.

Obrázek 4.7 Implementace čtečky QR kódů



Zdroj: vlastní zpracování

Třetí záložka v hlavní části

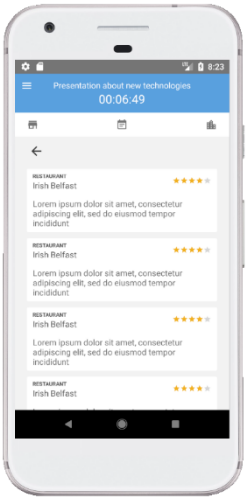
Zatím jsou v hlavní části pouze dvě záložky, kde první zobrazuje základní informace o nejbližších událostech a odpočet do následující události a druhá pak kalendářním zobrazením jednotlivé události.

Aby byla aplikace zajímavější, vznikl návrh implementovat třetí záložku, která bude obsahovat seznam kategorií míst. Každá kategorie obsahuje jednotlivá místa. Do kategorie se uživatel dostane stisknutím příslušné karty.

Je důležité zdůraznit, že se pro tyto účely používá stejná karetní komponenta, jako je tomu v případě karty události. Stejná komponenta je rovněž využita pro zobrazení jednotlivých míst a jejich hodnocení, které je vyjádřeno prostřednictvím zlatých hvězd, a tato implementace je zobrazena na obrázku 4.8.

Toto využití karet jakožto znovupoužitelné komponenty je jednou z výhod při tvorbě aplikací pomocí frameworku React Native. Výhoda je zejména ve zmenšení velikosti aplikace a zvýšení její přehlednosti.

Obrázek 4.8 Implementace hlavní části – třetí záložky, detailu kategorie



Zdroj: vlastní zpracování

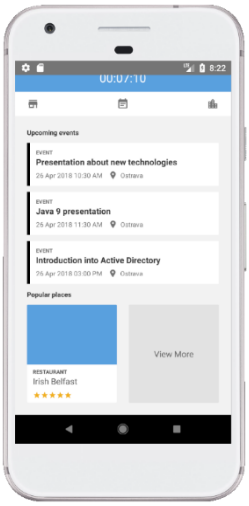
Karty míst v první záložce hlavní části

Tato část je jednou z těch jednodušších v rámci tvorby této aplikace. Došlo k využití komponenty `Flatlist` z React Native. Ten byl nastaven na horizontální polohu a povoleno jeho rolování. To je v zásadě vše, co bylo nutné k vytvoření jednoduché rolovací komponenty.

Tato komponenta vykresluje karty jednotlivých míst, vychází opět z vlastní karty a dovoluje, aby bylo možno zobrazit kartu bez obrázku a s hodnocením ve formě zlatých hvězd.

Při stisku na kartu představující místo je uživatel přesměrován na detail místa, který je popsaný v následující části této práce. Jako poslední karta v rolovací komponentě je šedá karta, která značí, že nelze zobrazit další karty míst. Implementace je zobrazena na obrázku 4.9.

Obrázek 4.9 Přidání míst do první záložky.



Zdroj: vlastní zpracování

Detail místa

Detail místa vychází z detailu události. V detailu místa jsou zobrazeny základní informace rozšířené zejména o umístění a popis. Jsou zde rovněž zobrazeny recenze ostatních návštěvníků, avšak z důvodu anonymizace nejsou tito návštěvníci prezentováni svými jmény.

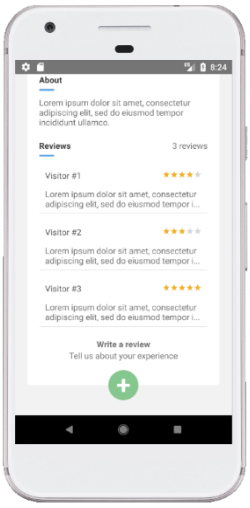
Je zde umístěna i vlastní komponenta vycházející z komponenty `Button` z React Native, která po stisknutí přesměrovává na formulář pro vytvoření vlastní recenze na dané místo.

Pro hodnocení je zde využita komponenta `TextInput` z React Native, která byla parametrizována na požadavky zejména v oblasti maximálního počtu zadaných znaků nebo počet řádků, a vlastní komponenta, která na základě zaslaných možností zobrazuje počet hvězd, název a indikátor. Při stisknutí jedné z možností dojde k uložení možnosti do stavového kontejneru.

Na základě toho, zdali je název možnosti ekvivalentní aktuálně zvolené možnosti, bude změněna barva indikátoru, aby bylo uživateli jasné, která varianta je zvolena. Po stisknutí tlačítka pro odeslání dojde k zaslání požadavku obsahující recenzi uživatele a dané místo.

Implementace detailu místa, tedy zejména část, která se liší oproti detailu události, je zobrazena na obrázku 4.10.

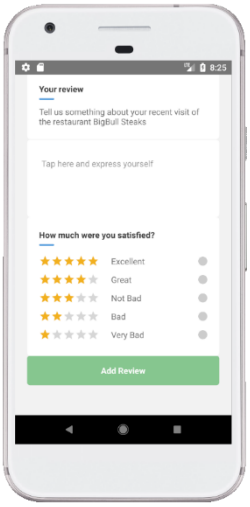
Obrázek 4.10 Implementace detailu místa



Zdroj: vlastní zpracování

Na obrázku 4.11 je zobrazena část pro vytvoření recenze místa.

Obrázek 4.11 Implementace recenzí míst



Zdroj: vlastní zpracování

5 Závěr

V rámci této práce došlo v druhé kapitole k popsání základních teoretických pojmů, které souvisejí s vývojem aplikaci pro platformu Android. Také byla popsána východiska jejich tvorby, včetně možností, jak lze mobilní aplikace pro platformu Android tvořit a jaké softwarové nástroje jsou užívány při jejich vývoji.

Znalost těchto pojmů a východisek byla základním pilířem pro celou práci a pochopení obecných principů, na základě kterých aplikace na platformě Android fungují, jsou nutností pro každého, kdo by se touto problematikou zabýval.

Následně byla ve třetí kapitole představena společnost Tieto s.r.o, ve které byla práce realizována. Také byl analyzován problém, který byl v rámci této práce řešen.

Ve čtvrté kapitole byla aplikace navržena s důrazem na její intuitivnost a příjemnost užívání. Dále bylo v této kapitole nutné pomocí analýzy zvolit optimální způsob pro tvorbu této aplikace. Byla představena kritéria pro její tvorbu a jejich výpočet pomocí Saatyho metody. Způsoby tvorby mobilních aplikací pro platformu Android popsané v druhé kapitole byly využity jako varianty při realizace této analýzy, která byla realizována s využitím metody PROMETHEE.

Na základě obdržených výsledků byl zvolen ideální způsob pro tvorbu aplikace, kterým je React Native. Tento optimální způsob byl podrobněji představen a spolu s ním i rozhodnutí, jakým způsobem řešit problémy při tvorbě aplikace při využití výše uvedeného způsobu tvorby aplikace.

Po představení zásadních otázek spojených s implementací aplikace došlo k vytvoření jejich jednotlivých částí. Implementace aplikace byla úspěšně dokončena a je připravena na nasazení v běžném provozu.

Hlavní cíl této práce, tedy analýza, návrh a vytvoření mobilní aplikace, která by kalendářním způsobem zobrazovala data získaná ze serveru, byl tedy rovněž splněn.

Navíc došlo i k implementaci vybraných vylepšení této aplikace oproti jejímu původnímu návrhu, avšak do produkční verze aplikace budou tato vylepšení přidána až po dokončení testování její původní verze. Poté bude implementováno řešení těchto vylepšení i na serverové straně a aplikace bude znovu testována. Následně bude aplikace uvedena do rutinního provozu a bude sloužit jako reprezentativní prvek společnosti v oblasti komunikace v rámci společnosti i se zákazníkem.

Využití frameworku React Native bylo doporučeno na základě analýzy a ukázalo se jako správná volba. Jedná se o novou technologii a s ní spojené problémy se však podařilo

vyřešit a do budoucna lze očekávat, že mnoho aplikací na mobilní zařízení bude tvořeno právě tímto způsobem.

Seznam použité literatury

Knihy

[1] BUCHALCEVOVÁ, Alena a Iva STANOVSKÁ. Příklady modelů analýzy a návrhu aplikace v UML. Praha: Oeconomica, 2013. 197 s. ISBN 978-80-245-1922-7.

[2] LACKO, Ľuboslav. Mistrovství - Android. Brno: Computer Press, 2017. 647 s. ISBN 978-80-251-4875-4.

[3] LACKO, Ľuboslav. Vývoj aplikací pro Android. Brno: Computer Press, 2015. 472 s. ISBN 978-80-251-4347-6.

[4] MU, Enrique a PEREYRA-ROJAS Milagros. *Practical Decision Making.* Cham (Švýcarsko): Springer International Publishing, 2016. 111 s. ISBN 978-3-319-33860-6.

[5] SCHILDT, Herbert. Java 8: výukový kurz. Brno: Computer Press, 2016. 696 s. ISBN 978-80-251-4665-1.

[6] VÁVRŮ, Jiří a Miroslav UJBÁNYAI. Programujeme pro Android. 2. rozš. vyd. Praha: Grada Publishing, 2013. ISBN 978-80-247-4863-4.

Články

[1] SAATY, Rozann Whitaker The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling* [online]. 1987, vol. 9, iss. 3-5, s. 161-176 [cit, 2018-04-19]. ISSN 0270-0255. Dostupné z: https://www.sciencedirect.com/science/article/pii/0270025587904738

Internetové zdroje

[1] Adobe, Inc., [online]. [cit. 2018-04-20]. Dostupné z: www.adobe.com

[2] Android Authority, [online]. [cit. 2018-04-27]. Dostupné z: https://www.androidauthority.com/history-android-os-name-789433/

[3] Android Developer - Activity Lifecycle, [online]. [cit. 2018-02-12]. Dostupné z: https://developer.android.com/

[4] Bizagi, Ltd., [online]. [cit. 2018-02-7]. Dostupné z: www.bizagi.com

[5] Discover SDK, [online]. [cit. 2018-03-02]. Dostupné z: http://www.discoversdk.com/blog/how-react-native-works

[6] Facebook, Inc., [online]. [cit. 2018-03-12]. Dostupné z: https://code.facebook.com/posts/597378980427792/react-native-a-year-in-review/

[7] Genymotion, [online]. [cit. 2018-04-27]. Dostupné z: https://www.genymotion.com/

[8] Github - React Native, [online]. [cit. 2018-03-12]. Dostupné z: https://github.com/facebook/react-native

[9] Github - React router navigation, [online]. [cit. 2018-03-12]. Dostupné z: https://github.com/LeoLeBras/react-router-navigation/

[10] Github Facebook, [online]. [cit. 2018-04-06]. Dostupné z: https://facebook.github.io/react-native/

[11] IDC, Inc., [online]. [cit. 2018-02-12]. Dostupné z: https://www.idc.com/promo/smartphone-market-share/os

[12] JetBrains, s.r.o. - Idea, [online]. [cit. 2018-02-16]. Dostupné z: https://www.jetbrains.com/

[13] Make Hybrid Apps, [online]. [cit. 2018-03-22]. Dostupné z: http://www.makehybridapps.com/2014/06/09/cordova-vs-phonegap-the-differences-and-which-one-to-use/

[14] Moment, [online]. [cit. 2018-04-12]. Dostupné z: www.momentjs.com

[15] Promethee Gaia, [online]. [cit. 2018-04-10]. Dostupné z: http://www.promethee-gaia.net/faq-pro/index.php?action=article&cat_id=003002&id=4

[16] Redux Saga, [online]. [cit. 2018-04-27]. Dostupné z: https://redux-saga.js.org

[17] Redux, [online]. [cit. 2018-03-27]. Dostupné z: https://redux.js.org/docs/advanced/AsyncActions.html

[18] Salesforce.com, Inc., [online]. [cit. 2018-03-27]. Dostupné z: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

[19] SimpleProgrammer, [online]. [cit. 2018-03-24]. Dostupné z: https://simpleprogrammer.com/progressive-web-applications/

[20] Statista, Inc., [online]. [cit. 2018-02-02]. Dostupné z: https://www.statista.com/statistics/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system/

[21] Techadvisor, [online]. [cit. 2018-03-28]. Dostupné z: https://www.techadvisor.co.uk/download/operating-systems-distros/microsoft-visual-studio-emulator-android-10507154-3330516/

[22] Tutorialspoint, [online]. [cit. 2018-02-16]. Dostupné z: https://www.tutorialspoint.com/eclipse/eclipse_overview.htm

[23] Xenonstack, [online]. [cit. 2018-03-12]. Dostupné z: https://www.xenonstack.com/blog/updates/overview-of-kotlin-comparison-between-kotlin-java

Seznam zkratek

DEX	Dalvik Executable
HTC	High Tech Computer
HTML	Hyper Text Markup Language
Inc	Incorporated
JAR	Java Archive
JSX	JavaScript Extensible Markup Language
Ltd.	Limited
NDK	Native Development Kit
PROMETHEE	Preference Ranking Organization Method for Enrichment of Evaluations
PWA	Progressive Web Applications
Q	Čtvrtletí
QR	Quick Response
SDK	Standard Development Kit
SMS	Short Message Service
s.r.o.	společnost s ručením omezeným
UC	Případ užití
XML	Extensible Markup Language

Prohlášení o využití výsledků bakalářské práce

Prohlašuji, že

- jsem byl seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, bakalářskou práci užít (§ 35 odst. 3);
- souhlasím s tím, že bakalářská práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího bakalářské práce. Souhlasím s tím, že bibliografické údaje o bakalářské práci budou zveřejněny v informačním systému VŠB-TUO;
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;
- bylo sjednáno, že užít své dílo, bakalářskou práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne 11.5.2017

.....

Jan Hošna

Seznam příloh